SAIC-95/1029

Global Association System Phase 2: Conflict Resolution

Design Document

G. Beall, R. Le Bras, W. Nagy, T. Sereno and H. Swanger

February 22, 1995

Science Applications International Corporation
10260 Campus Point Drive
San Diego, California 92121

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 22 February 1995	3. REPORT TYPE AND DATES COVERED Technical Report Dec 94 - Mar 95

4. TITLE AND SUBTITLE Global Association System Phase 2: Conflict Resolution	5. FUNDING NUMBERS F08606-90-D-0005
6. AUTHOR(S) G. Beall, R. LeBras, W. Nagy, T. Sereno and H. Swanger	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Science Applications International Corporation 10260 Campus Pt. Drive San Diego, CA 92121	8. PERFORMING ORGANIZATION REPORT NUMBER SAIC ~~SAOC~~-95/1029
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ Air Force Technical Applications Center (HQ AFTAC/TTR) 1030 S. Highway A1A Patrick AFB FL 32925-3002	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release, Distribution Unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This document describes the second phase in the development of a *Global Association System* to perform automatic interpretation of seismic data to associate signals from a network of stations and locate seismic events using a method similar to generalized beam forming [e.g., *Ringdal and Kværna*, 1989; *Taylor and Leonard*, 1992; *Leonard*, 1993]. The major improvement over the first phase is the implementation of a new method to resolve conflicts among phases that are associated with more than one preliminary event hypothesis. This task was performed by a separate expert system called **ESAL** in the first phase [*Le Bras et al., 1994a*]. This document also describes our design for the third phase of the *Global Association System* which will include the association of non-defining secondary phases and late-arriving data.

14. SUBJECT TERMS Global Association; GA; Conflict Resolution; Event Identification; Design Document; SAIC-95/1029	15. NUMBER OF PAGES ~~46~~ 47
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	Same as Report

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z39-18
298-102

(This page intentionally left blank.)

Table of Contents

Page

List of Figures

List of Tables

1.0 Introduction

This document describes the second phase in the development of a *Global Association System* to perform automatic interpretation of seismic data to associate signals from a network of stations and locate seismic events using a method similar to generalized beam forming [e.g., *Ringdal and Kværna*, 1989; *Taylor and Leonard*, 1992; *Leonard*, 1993]. The major improvement over the first phase is the implementation of a new method to resolve conflicts among phases that are associated with more than one preliminary event hypothesis. This task was performed by a separate expert system called **ESAL** in the first phase [*Le Bras et al., 1994a*]. This document also describes our design for the third phase of the *Global Association System* which will include the association of non-defining secondary phases and late-arriving data.

1.1 Background

In 1993, discussions began concerning the design of the International Data Center (IDC) and the U.S. National Data Center (NDC) for the upcoming Group of Scientific Experts Third Technical Test, called GSETT-3 (for an overview, see *Kerr* [1993]). The GSETT-3 experiment will be the first demonstration of a global monitoring system that addresses the CTBT problem. In the early discussions, the global network was envisioned to include as many as 60 primary stations (mostly arrays) to provide continuous data, and up to 200 auxiliary stations to provide waveform segments upon request. The volume of data (~10 Gbytes per day) and number of events (300-400 per day) were estimated to be approximately a factor of five to ten times greater than encountered from existing global networks.

SAIC performed an engineering study to assess whether existing seismic monitoring systems could be modified to handle these expected data volumes. The software components of ADSN and IMS were considered. The conclusion of this study was that the primary bottleneck would be the automatic association and location program, **ESAL** [*Bratt et al.*, 1991, 1994]. Performance analyses indicated that **ESAL**'s execution time scaled roughly with the square of the detection density. This was considered unacceptable since extrapolation to the estimated data volumes for a CTBT monitoring network indicated that **ESAL** would not be able to process the data in real time.

To address this concern, SAIC proposed a phased replacement of **ESAL** with a new *Global Association System* that would allow some of the tasks to be run in parallel:

- **Phase 1:** Station processing and preliminary event formation were replaced with new modules called **StaPro**, **GAcons** and **GAassoc**. This hybrid system uses a method similar to generalized beam forming for automatic association and event location [e.g., *Ringdal and Kværna*, 1989; *Taylor and Leonard*, 1992; *Leonard*, 1993]. Conflict resolution and event refinement are performed by **ESAL** [*Le Bras et al.*, 1994*a*].
- **Phase 2:** Conflict resolution was moved into the *Global Association System*. **ESAL** still associates secondary phases and late-arriving data. Shared modules for conflict resolution were developed and implemented into **GAassoc** and **GAconflict**.
- **Phase 3:** This completes the replacement of **ESAL** by implementing the association of secondary phases and late-arriving data in the *Global Association System*. This document describes a design for this phase which has not yet been implemented (see Section 5.0).

1.2 Report Outline

Section 2 gives a high-level description of Phase 2 of the *Global Association System.* Sections 3 and 4 describe our approach to conflict resolution and its implementation. Detailed descriptions of the algorithms and diagrams representing the data flow are included. Section 5 describes the design for Phase 3 which incorporates association of secondary phases and late-arriving data.

2.0 System Summary

This section provides a brief description of the main components of Phase 2 of the *Global Association System.*

- **StaPro** - This module performs station processing. It analyzes detections and their features to make preliminary seismic phase identifications. **StaPro** contains the same logic as **ESAL** for this task, and it adds the ability to compute single-station location and magnitude hypotheses. This information is used by **GAassoc** to screen detections from local events with magnitudes less than a user-specified threshold. This module is considerably more compact than **ESAL**, permitting each station to be processed independently and in parallel.

- **GAcons** - This module builds a global grid file containing the knowledge base for the association process. Overlapping circular grid cells provide complete global coverage, including depth cells in areas where deep seismicity is known to occur. The information contained in the grid file includes travel time, slowness, and azimuth bounds, and information on the probability of detection for each station in the network. A graphical user-interface is available to review and edit the grid values.

- **GAassoc** - This module identifies event hypotheses using an exhaustive search over all grid cells. It uses the information in the grid file produced by **GAcons** to identify detections that are consistent with a particular event hypothesis. Multiple instances of **GAassoc** can be run in parallel with each instance forming event hypotheses for a different sector of the Earth. The preliminary bulletin produced by **GAassoc** will not contain conflicting associations (i.e., phases that are associated with more than one event hypothesis) within any sector, but can contain conflicting associations between sectors and with previously processed time intervals. These conflicts are resolved by **GAconflict**.

- **GAconflict** - This module resolves conflicts between events formed in different sectors in the current time interval by different instances of **GAassoc** and between the events formed in the current time interval and events formed in previous time intervals.

- **EServer/ESAL** - These modules are used in Phase 2 to refine the event hypotheses formed by the *Global Association System* by associating secondary phases and late-arriving data. **EServer** is a data agent for **ESAL** that prepares ASCII input files from data read from a commercial relational database management system (RDBMS). **EServer** also writes events and associations determined by **ESAL** to the RDBMS. When Phase 3 of the system is implemented, these functions of **ESAL** will be implemented within the *Global Association System.*

Figure 1 is a schematic view of Phase 2 of the *Global Association System.* **StaPro**, **GAcons**, and **EServer/ESAL** have not changed much since Phase 1, and their roles are described by *Le Bras, et al.* [1994*a*]. Enhancements to **GAassoc** and the new module called **GAconflict** are described in detail in Sections 3 and 4.

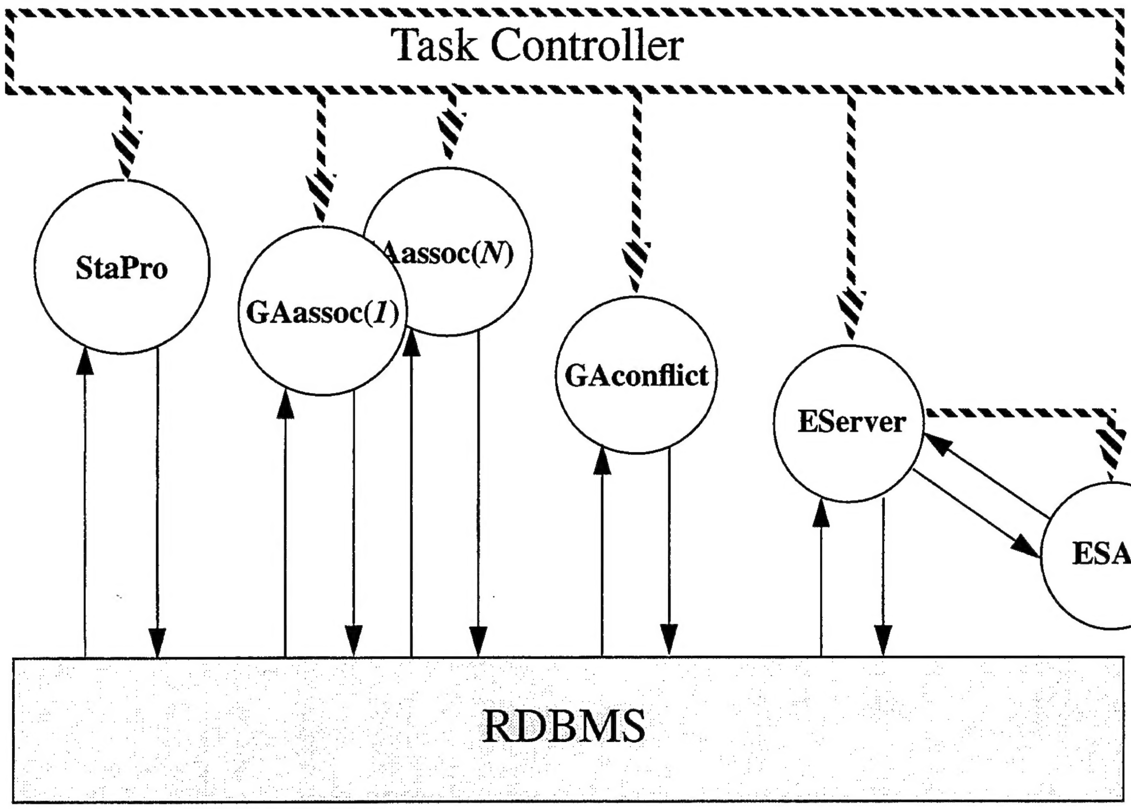

Figure 1. This figure shows the high-level processing and data flow for Phase 2 of the *Global Association System*. Process flow is indicated by dashed lines and data flow is indicated by solid lines. Note the parallel instances of **GAassoc.**

The *Global Association System* was developed on UNIX workstations under the Solaris 2.3 operating system using the C programming language (ANSI-compatible). The current version uses an Oracle^{TM} 7.1.3 RDBMS and can be ported to other commercial databases that are supported by the Generic Database Interface (GDI) developed by SAIC [*Anderson et al.*, 1994]. **StaPro** uses CLIPS Version 6.0 to provide run-time configurability of station-specific rules. CLIPS is a knowledge-based macro language which is supported by NASA. **ESAL** is programmed in the ART (Automated Reasoning Tool) expert system shell from Inference Corporation.

3.0 Conflict Resolution Overview

3.1 Algorithm

This section is an overview of the implementation of conflict resolution in the *Global Association System.* An arrival is in conflict when it is associated with more than one preliminary event hypothesis. Conflict resolution is performed in both **GAassoc** and **GAconflict. GAassoc** may process data from the whole Earth in a single instance, or it may process data from different sectors of the Earth in separate instances running in parallel. In either case, it will resolve conflicts that arise during processing of a single sector. Conflicts can also arise between sectors or between event hypotheses from the current time interval and previous time intervals. These conflicts are resolved by **GAconflict.** Their roles are illustrated in Figure 2.

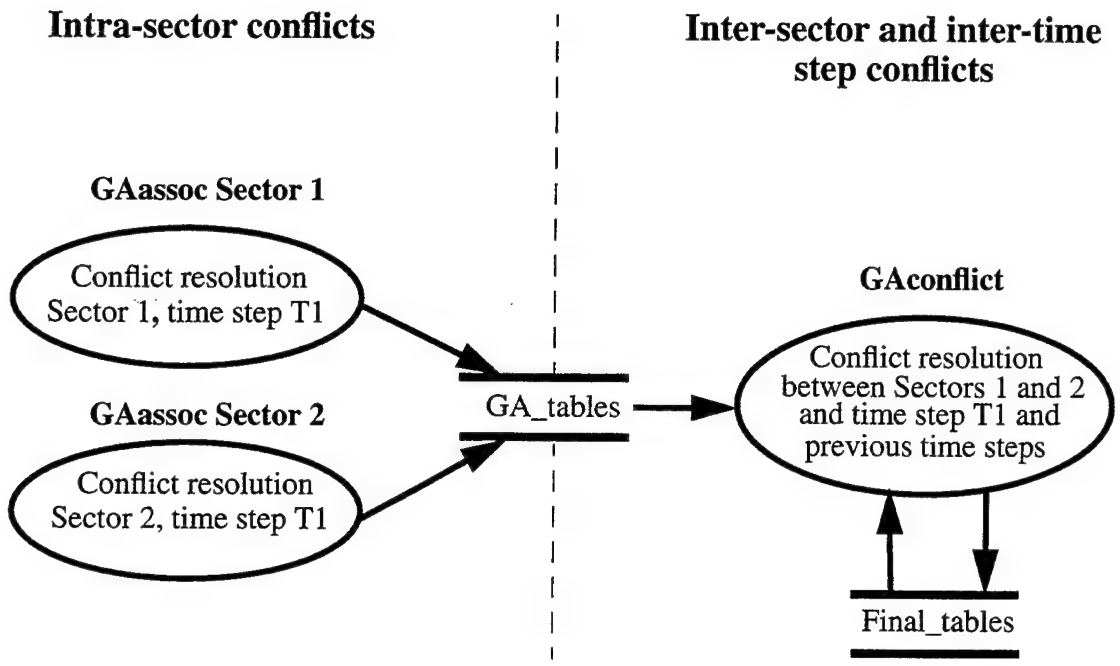
Intra-sector conflicts
Inter-sector and inter-time step conflicts
GAassoc Sector 1
Conflict resolution Sector 1, time step T1
GAassoc Sector 2
Conflict resolution Sector 2, time step T1
GA_tables
GAconflict
Conflict resolution between Sectors 1 and 2 and time step T1 and previous time steps
Final_tables

Figure 2. This figure illustrates the respective roles of **GAassoc** and **GAconflict** in the resolution of conflicts within the *Global Association System.* **GAassoc** resolves conflicts that arise during the processing of a single sector and time step. All instances of **GAassoc** write to the same database tables. **GAconflict** reads from those tables and the final event tables and resolves both conflicts between sectors and those that arise between the current time step and previous time steps.

GAassoc and **GAconflict** use the same two methods to resolve conflicts. The first method uses a clustering algorithm to identify and resolve conflicts between large events that share a high percentage of associated arrivals. Remaining conflicts are resolved by constructing a metric to rank the events that associate the same arrival. This metric includes a measure of the quality of the event solution and the goodness-of-fit of each association.

3.2 Data Flow Diagrams

This section presents the data flow diagrams (DFDs) for three main components of the *Global Association System*: **GAcons**, **GAassoc** and **GAconflict**. At any level, the DFD provides a view of the system as an interaction between modules. The bubbles represent functional elements of the system at the specific level of decomposition. The solid arrows indicate data exchange between modules and data stores, which are indicated by heavy horizontal lines. Data stores include database tables, external data files, and internal data structures. The crosshatched arrows indicate data input by the user. Important subroutines called within the various modules are noted between square brackets.

The lower-level data flow diagrams for **GAcons** have not changed from Phase 1 so they are not included in this report [see *Le Bras et al.*, 1994*a*]. The diagrams for **GAassoc,** which have been modified to include the new modules for resolving conflicts, are presented in Section 3.2.2. The new **GAconflict** program uses the same modules as **GAassoc** to resolve conflicts, so only the top-level data flow diagram is included in Section 3.2.3.

3.2.1 Top Level GA Diagram

The Level 1 Data Flow Diagram (DFD) for the *GA Subsystem* is shown in Figure 3. It shows the interaction of the three main programs, **GAcons**, **GAassoc** and **GAconflict**.

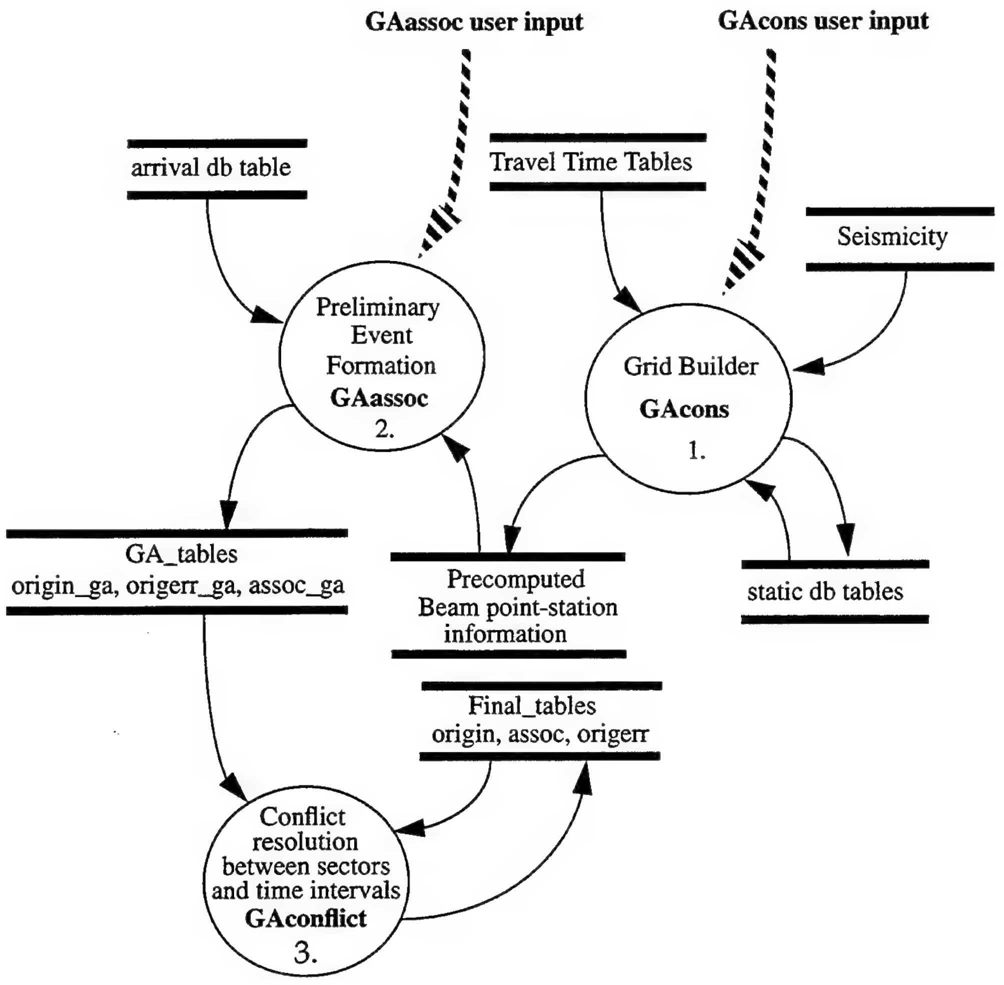

Figure 3. Level 1 DFD for the *GA Subsystem*. The three functional units at this level are **GAcons, GAassoc** and **GAconflict.**

GAcons produces one or several grid files that contain the knowledge base used by **GAassoc** to perform the automatic association of arrivals. It only needs to be run when the network or grid characterization changes and is not part of the real-time processing system.

GAassoc uses arrival data for the current time interval and the grid information produced by **GAcons** to generate self-consistent sets of associations. Sets that pass various acceptance tests and the process of conflict resolution become preliminary events. These are written to an intermediate set of tables that we call the *GA_tables.* Several instances of **GAassoc** may be run in parallel, each on a different sector or region of the Earth. Each instance writes its results to the same intermediate set of tables.

GAconflict resolves conflicts in the *GA_tables* that arise from parallel processing of multiple sectors. Results are written to a separate set of database tables which we call the *Final_tables*. **GAconflict** also reads the *Final_tables* to identify conflicts between the new events in the *GA_tables* and events from earlier time intervals, which are in the *Final_tables*. The events that are in conflict with a new event are removed from the *Final_tables,* and they are reprocessed by **GAconflict** with the events from the *GA_tables*. The *Final_tables* do not contain any conflicting associations.

3.2.2 Data Flow Diagrams for GAassoc

The three data flow diagrams for **GAassoc** illustrate the modular structure of the program. Most modules use the same data structure, called the *Driver* structure (see Figures 4, 5, and 6). Figure 4 is a high-level view of **GAassoc**. The functional units are (2.1) preliminary association, and (2.2) event location and conflict resolution.

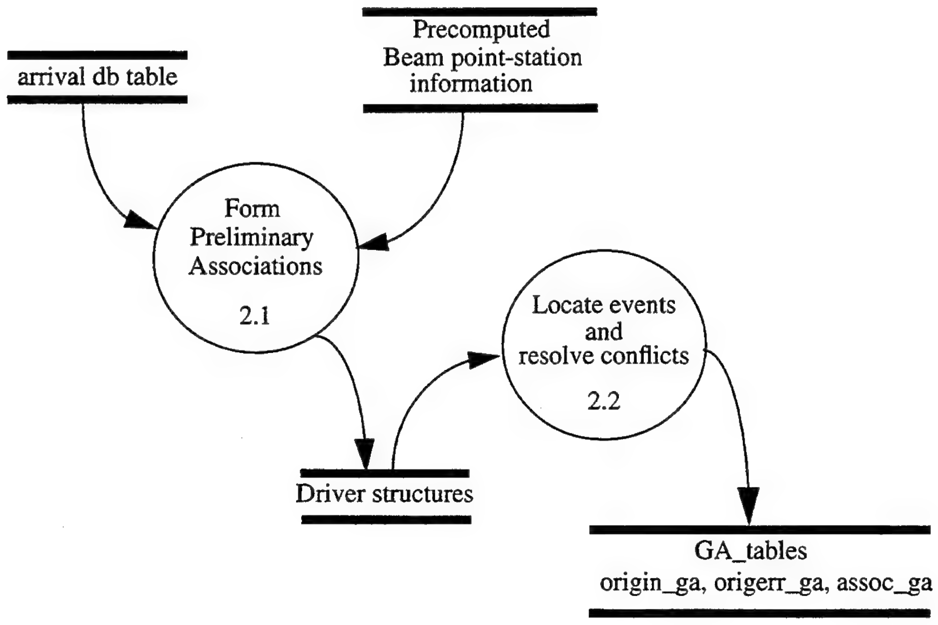
arrival db table
Precomputed Beam point-station information
Form Preliminary Associations 2.1
Locate events and resolve conflicts 2.2
Driver structures
GA_tables origin_ga, origerr_ga, assoc_ga

Figure 4. Level 2 DFD for **GAassoc**. The main functions of **GAassoc** are to form preliminary associations (2.1) and to locate events and resolve conflicts (2.2).

Figure 5 is a detailed expansion of the first part of **GAassoc** where data are read and preliminary associations are formed.

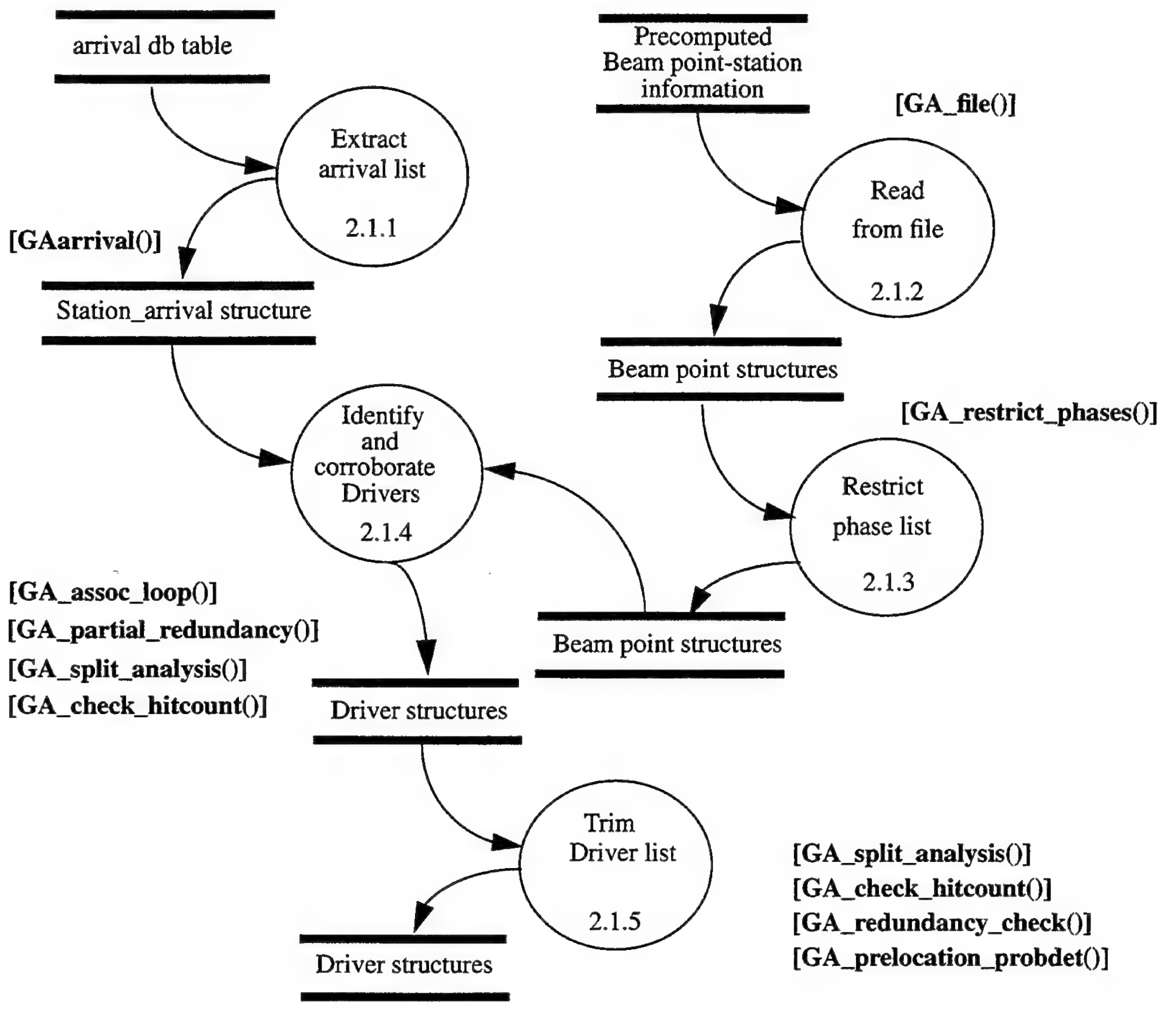

Figure 5. Level 3 DFD diagram for **GAassoc**. Preliminary events are formed by associating the arrivals read from the database. Travel time, slowness and amplitude information stored in the Beam point structures are used to identify associations.

Figure 6 is a detailed expansion of the second part of **GAassoc** where events are located and conflicts are resolved.

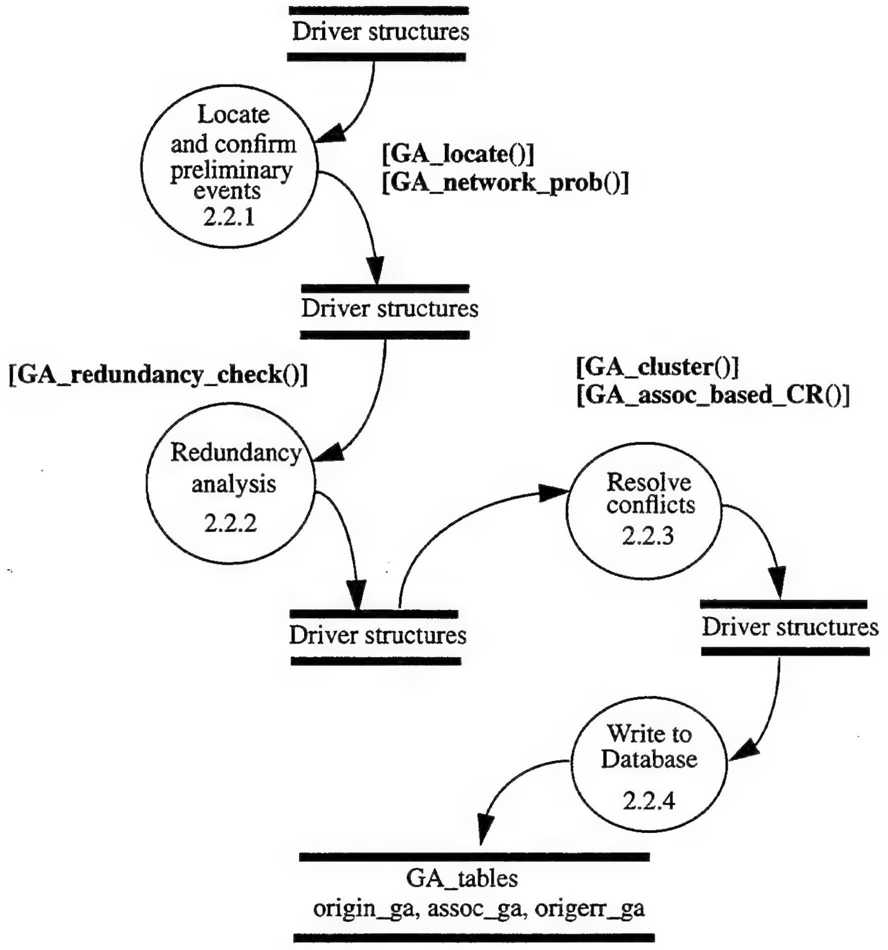

Figure 6. Level 3 DFD diagram for **GAassoc**. Once preliminary events are formed, they are located, analyzed for redundancy and conflicts between them are resolved. Conflict resolution utilizes two distinct methods: event clustering and association-based resolution (see Sections 4.1 and 4.2 for detailed descriptions).

3.2.3 Data Flow Diagrams for GAconflict

Figure 7 shows the data flow diagram for **GAconflict**. Note that **GAconflict** uses the same modules as **GAassoc** to perform redundancy analysis and conflict resolution.

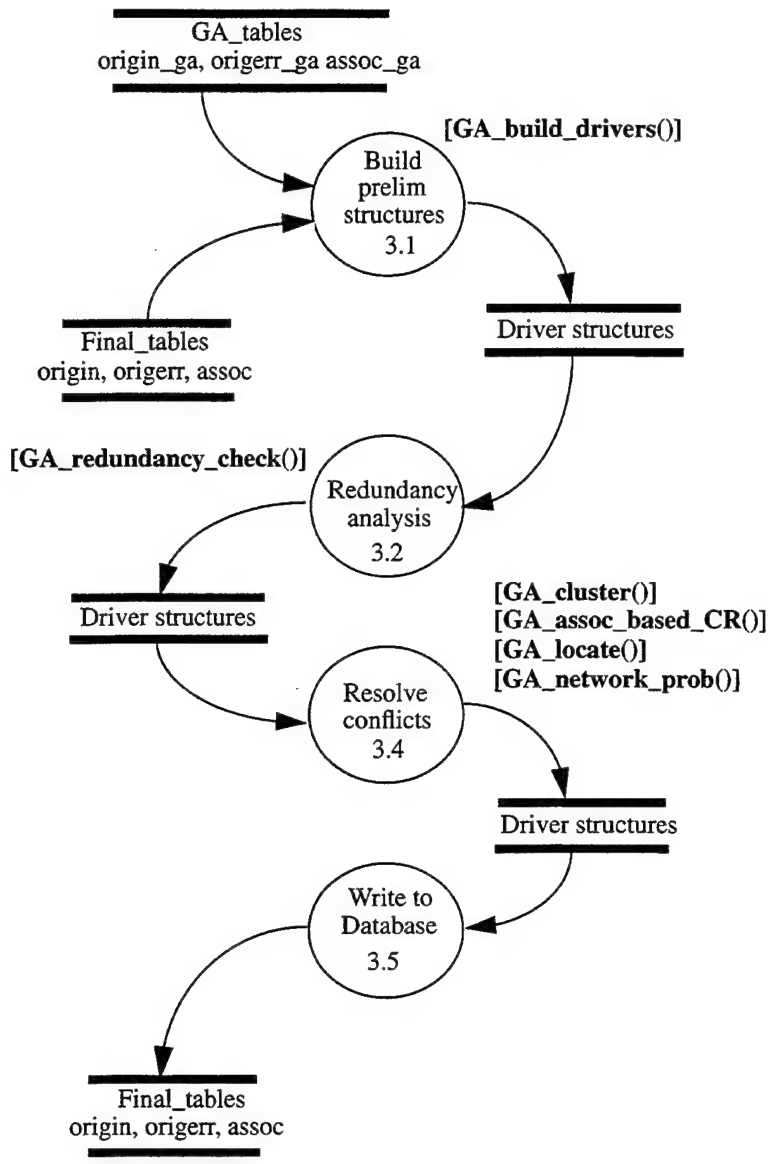

Figure 7. Level 2 DFD for **GAconflict**. **GAconflict** uses the same modules as **GAassoc** to perform redundancy analysis and conflict resolution.

4.0 Detailed Description of Shared Modules

This section describes the two methods used in the *Global Association System* for conflict resolution. The first method is a special purpose method which uses a clustering algorithm to identify and resolve conflicts between large events that share a high percentage of associated arrivals (Section 4.1). The second method is a general purpose method which uses an association-based measure to resolve the remaining conflicts on the basis of the quality of the event solution and the goodness-of-fit of each association (Section 4.2).

4.1 Cluster Analysis

The first method in conflict resolution uses a clustering technique to identify and resolve conflicts between large events that share a high percentage of associated arrivals. It was introduced to reduce the problem of splitting large events. **GAassoc** frequently generates numerous small variations of a large event. Experience has shown that association-based conflict resolution has difficulty distinguishing between these variations and tends to split a large event into multiple smaller events. Cluster analysis provides a means of identifying a group of similar preliminary events and reducing that group to the single "best event." We define the "best event" as the one with the largest number of defining phases; the size of the error ellipse is used to break ties. The clustering method is only applied to clusters of large events (i.e., many defining phases) so that we have a high-confidence that all members represent the same event. The application of this method is controlled by the user parameter *do_clustering*.

4.1.1 Algorithm

The key steps of the clustering algorithm are:

- Select the preliminary event with the largest number of defining phases that has not already been clustered. This number must be greater than or equal to a user-specified limit (*cluster_min_ndef*). If several events have the same number of defining arrivals, then select the one with the smallest error ellipse. Call this event the "best event."
- Form a cluster by identifying all other preliminary events that have at least a specified percentage (*cluster_min_pct_overlap*) of associations in common with the "best event.". These associations must be time-defining[1]. The phase identifications may be different for the different preliminary events.
- Dissolve all preliminary events in the cluster except for the "best event."
- Continue as long as there are preliminary events with a sufficient number of defining phases.

4.1.2 Major Software Components

The major software components for cluster analysis are summarized in Table 1.

Table 1: Major modules for cluster analysis

Subroutine Prototype	Short Description of Function	File name
int GA_cluster (Driver **dr_anch, int cluster_min_ndef, double cluster_min_pct_overlap	Performs event cluster analysis on preliminary events (or *Drivers*) with ≥ *cluster_min_ndef* defining data. We start with the event with the largest number of defining data and search through the remaining events to remove those containing association sets with at least *cluster_min_pct_overlap* of the smaller set.	**GA_cluster.c**

1. Time-defining means that the arrival time was used in the calculation of the event location.

4.2 Association-Based Conflict Resolution

Conflicts that remain after cluster analysis are resolved by an association-based method. Each arrival that is associated with more than one event hypothesis is assigned to the event that maximizes a weighted product of the goodness-of-fit and a measure of the quality of the event solution. The goodness-of-fit is based on time, azimuth, slowness and log amplitude residuals. The event quality is based on the number of defining observations, size of the error ellipse, distance to the nearest station, and probability of detection, plus a factor to help retain small events. The test is applied iteratively and all events are relocated and all measures recomputed after each disassociation of an arrival. The application of this method is controlled by the user parameter *do_association_based_conflict_resolution.*

4.2.1 Algorithm

The conflicting arrival is assigned to the event hypothesis that maximizes the quality measure:

$$L_{ij} = F_{ij}^{a} \cdot Q_i^{(1-a)} \tag{1}$$

where F_{ij} is a measure of the goodness-of-fit of the jth arrival to the ith event, Q_i is a measure of the quality of the ith event, and a is a user-specified weighting factor (*master_tradeoff_weight*). If a is zero then the goodness-of-fit of the arrival to the event solution will be ignored, and conflicts will be resolved in favor of the event with the highest quality. Conversely, if a is set to one then conflicts will be resolved only on the basis of goodness-of-fit. Both F_{ij} and Q_i are normalized between 0 and 1, so L_{ij} also varies in this range.

Goodness-of-Fit

The goodness-of-fit of an association, F_{ij}, is based on the time, azimuth, slowness, and log amplitude residuals when available. A χ^2 value is computed as:

$$\chi^2 = \sum_j ((d_j - m_j)/\sigma_j)^2 \tag{2}$$

where d_j is the observed data, m_j is the theoretical data and σ_j is the estimated standard deviation of the jth datum. The χ^2 is unbiased by using the equation:

$$\chi^2 = \chi^2 \cdot Ntot/(Ntot - Ndata) \tag{3}$$

where *Ntot* is the total number of data used to compute location and magnitude in the event hypothesis minus the number of model parameters in location and magnitude, and *Ndata* is the number of data for the arrival in question. The quantity F_{ij} is the probability corresponding to this unbiased χ^2 value.

Event Quality

The quality of an event, Q_i, is the normalized weighted sum of two terms:

$$Q_i = (b \cdot Q_{1i} + c \cdot Q_{2i}) / (b + c) \tag{4}$$

The first term is a measure of the quality of the event solution based on the number of defining phases, the size of the error ellipse, the distance to the nearest station, and the probability of detection. The second term increases the event quality measure if it is likely that the event would be dissolved if the association is removed from it (i.e., if the number of defining phases is small). The user-specified factors *b* (*event_likelihood_weight*) and *c* (*dissolved_event_weight*) allow adjustment of the relative weight given to these two terms.

The first term, Q_1, is a measure of how likely the event is to be real. It is computed as the following normalized weighted sum:

$$Q_1 = \sum_{i=1}^{NA} w_i \cdot M_i \Big/ \sum_{i=1}^{NA} w_i \tag{5}$$

where *NA* is the number of event quality attributes, w_i is the user-specified weight assigned to each attribute, and M_i is a measure between zero and one of how likely the event is to be real based only on the ith attribute. The M_i are approximated as linear ramps from 0.0 to 1.0 between interval bounds that are specified for each attribute (Figure 8).

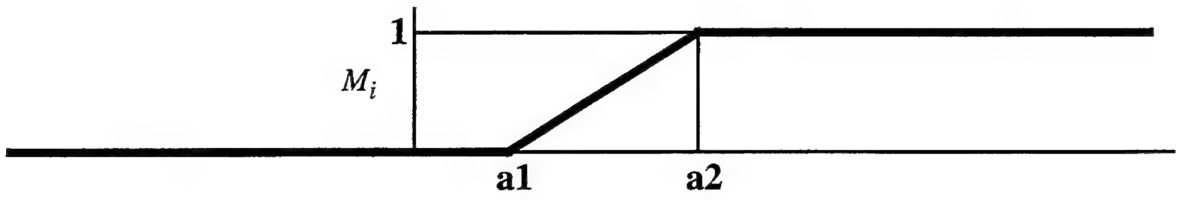

Figure 8. This figure shows the shape of the function M_i. The interval bounds, *a1* and *a2,* and the sign of the slope completely define M_i for each attribute.

The event quality attributes used to compute Q_1 are:

1 - The number of defining phases (ndef). If an event has a large number of defining phases then it is more likely to be real than if it has a small number of defining phases. The interval bounds for M_i are specified by the user-parameters: *ndef_no_confidence_bound* (default is 3) and *ndef_high_confidence_bound* (default is 10). The weight is specified by the user parameter *ndef_weight* (default is 1.0).

2 - The size of the error ellipse. If the error ellipse is small then there is reasonable network coverage and the event is more likely to be real than if the error ellipse is large. The interval bounds for M_i are specified by the user-parameters: *smajax_no_confidence_bound* (default is 500 km) and *smajax_high_confidence_bound* (default is 10 km). The weight is specified by the user parameter *smajax_weight* (default is 0.8).

3 - The distance to the nearest station. If the nearest station is close, then the event is more likely to be real than if it is far. The interval bounds for M_i are specified by the user-parameters: *dnear_no_confidence_bound* (default is 90 km) and *dnear_high_confidence_bound* (default is 10 km). The weight is specified by the user parameter *dnear_weight* (default is 0.5).

4 - Probability of detection. If the network probability of detection estimate is consistent with the set of stations that detected the event, then the event is likely to be real. The interval bounds are applied to the ratio of the residual used in the probability of detection event confirmation test and its standard deviation [*Le Bras et al.*, 1994*a*]. They are specified by the user-parameters: *probdet_no_confidence_bound* (default is 3.0) and *probdet_high_confidence_bound* (default is 1.0). The weight is specified by the user parameter *probdet_weight* (default is 0.7).

An additional location-specific attribute, seismicity, is planned to be added to the above set. An event is more likely to be real if it is located in an area characterized by high seismic activity. The current program does not use this attribute.

The second term in Q_i, Q_2, is provided to reduce the possibility of dissolving a small event. It is defined in terms of the number of defining phases as:

Q_2 = 0.0 if the number of defining arrivals larger than or equal to *ndef_not_likely_to_dissolve_event* (default is 6).

Q_2 = 1.0 if the number of defining arrivals is less than or equal to *ndef_which_will_dissolve_event* (default is 3).

Q_2 is linear between these two bounds.

Procedure

The iterative procedure for applying the metrics described above to resolve conflicts is described in this section. It is initialized by computing Q_i for all events and L_{ij} for each of the conflicting associations. The following tasks are performed once for each event with conflicting associations:

- **Task 1:** Rank all conflicting associations based on their quality measure, L_{ij}. The rank will be set to zero for the event with the highest L_{ij}, and it will be greater than zero for all other events. Select the event with at least one conflicting association whose rank is zero that has the highest proportion of defining associations that it is likely to keep after conflict resolution. This proportion is $(n_1+n_2)/ndef$, where n_1 is the number of defining associations that are not in conflict, n_2 is the number of conflicting associations with rank equal zero, and *ndef* is the total number of defining associations for the event. Ties are broken by selecting the event with the highest event quality, Q_i.
- **Task 2:** For the current event, select the conflicting association with the highest quality measure, L_{ij}, and disassociate this arrival from all other events.
- **Task 3:** Relocate all events that have lost an association and reapply the event confirmation criteria. Abandon any events that no longer satisfy these criteria and disassociate all of their arrivals. Recompute Q_i and L_{ij} for all events that have lost an association and re-rank all affected associations. Return to Task 2 if the current event has remaining conflicting associations with rank equal zero.

Several iterations through this procedure may be required to resolve all conflicts. The example below shows such a case.

4.2.2 Example

Figures 9 and 10 illustrate a simple example of association-based conflict resolution. We start with four events with conflicting associations of three arrivals. Figure 9 shows the events, their associated arrivals, and the conflicts. The arrivals are labelled α, β and δ. We assume for simplicity in this example that the ranking of associations is not modified after events are relocated.

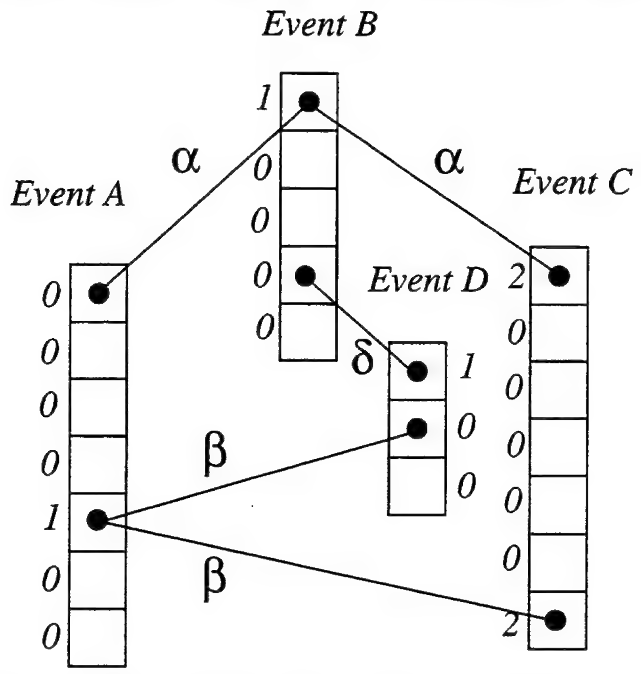

Figure 9. Illustration of association-based conflict resolution. Four events, *A, B, C* and *D,* have three associated arrivals in conflict, α, β and δ. Each square represents an arrival. The number next to the square is the rank of the association: 0 represents the best association for that arrival. The lines join the same arrival associated with different events.

Two iterations of the procedure described above are required to resolve all conflicts in this example. Figure 10 illustrates the results at intermediate stages. The following describes the first iteration, with tasks identified corresponding to the procedure described above:

- Task 1: Event *A* is selected.
- Task 2: Arrival α is disassociated from Events *B* and *C.*
- Task 3: Events *B* and *C* are relocated, and both still pass event confirmation (Figure 10a).

- Task 1: Event *B* is selected.
- Task 2: Arrival δ is disassociated from Event *D.*
- Task 3: Event *D* is abandoned because it does not satisfy event confirmation. Conflicting associations of arrival β are re-ranked (Figure 10b).

Event *C* is not selected in the first iteration because it does not have any conflicting arrivals with rank equal to zero. Event *D* is not selected because it is dissolved during the processes of resolving the conflicts with Event *B*. The second iteration must resolve only one conflict:

- Task 1: Event *A* is selected.
- Task 2: Arrival β is disassociated from Event *C*.
- Task 3: Event *C* is relocated (Figure 10c).

All conflicts are resolved after these two iterations and Events *A*, *B* and *C* remain. The re-ranking after each conflict is resolved is an important step in the procedure. In this example, arrival β would have been disassociated from Event *A* if the ranks were not updated.

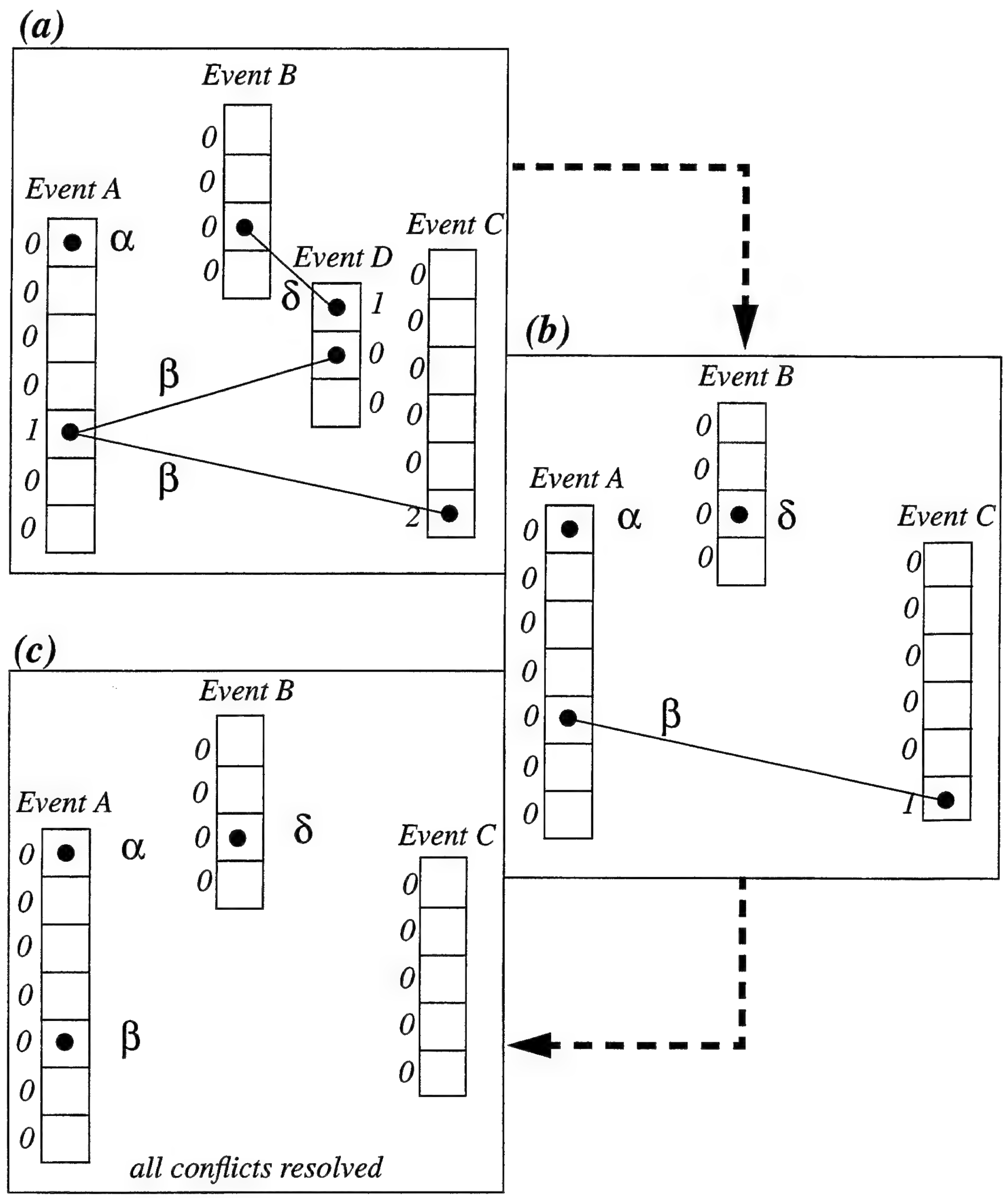

Figure 10. Illustration of the association-based conflict resolution procedure. (a) Event *A* is selected and conflicts for arrival α are resolved. (b) Event *B* is selected, and conflicts for arrival δ are resolved. Event *D* is dissolved during this process. (c) Event *A* is selected in the second iteration, and conflicts for arrival β are resolved. All conflicts are resolved and three events remain.

4.2.3 Major Software Components

The major software components for association-based conflict resolution are listed in Table 2.

Table 2: Major modules for association-based conflict resolution

This table lists the software modules for association-based conflict resolution with a short description of their function and the name of the file where they reside.

Subroutine Prototype	**Short description of function**	**File name**
int GA_assoc_based_CR (Driver **dr_anch, Ev_Confirm *ev_confirm, CR_params cr_params, Locator_params *loc_params, Site *sites, Netwrk *net_sta, int num_sites)	Performs association-based conflict resolution on preliminary events as controlled by various weighting values.	**GA_assoc_based_CR.c**
int GA_main_CR_loop (Driver **dr_anch, Driver *dr_cur, Ev_Confirm *ev_confirm, CR_params cr_params, Locator_params *loc_params, Site *sites, Netwrk *net_sta, int num_sites)	Performs Tasks 2 and 3 of the association-based conflict resolution procedure for each selected event.	**GA_assoc_based_CR.c**
double GA_event_quality (Driver *dr, CR_params cr_params)	Computes the event quality measure, Q_i.	**GA_assoc_based_CR.c**
void GA_arrival_quality (Driver **dr_anch, double exp_wt)	Computes the quality measure, L_{ij}.	**GA_assoc_based_CR.c**

Table 2 (cont.).

Subroutine Prototype.	**Short description of function.**	**File name.**
double GA_member_scaling (double sample_value, double low_conf_value, double high_conf_value, Bool larger_value_is_more_important, Bool log_scaling)	Scales a given input value according to user-specified confidence settings established in the structure, CR_params. This is a normalized scaling (i.e., 0.0 to 1.0).	**GA_assoc_based_CR.c**
void GA_rank_arrivals (Driver **dr_anch)	Ranks conflicting associations according to their conflict resolution quality measure: the best quality association will be in first index position, i.e., 0. This first entry has special significance by being the best fit, and is therefore, given the name, "winner". The remaining "losers" will follow indexed from 1 to number of conflicting associations. These "losers" can potentially become "winners", if some of the "winners" are in events which are subsequently dissolved.	**GA_assoc_based_CR.c**

4.3 Common Data Structures

The data structures used in Phase 2 of the *Global Association System* have been upgraded to satisfy the needs of the conflict resolution modules. This primarily involves the *Driver* data structure which is now used throughout the system, from the early stage of the association loop within **GAassoc** to the final stage of inter-sector and inter-time step conflict resolution within **GAconflict**. Table 3 shows the details of the *Driver* data structure and a related structure.

Table 3: Data structures used in GAassoc and GAconflict.

This list is limited to the *Driver* structure and the accessory *assoc_CR* structure, which are central to both **GAassoc** and **GAconflict**. Refer to *Le Bras et al.*[1994a] for details on other data structures.

Structure name and definition.	Short description of elements.	File name.
typedef struct driver Driver **{**	*Driver* or generator. This is a structure formed to contain preliminary event information. It contains pointers to static station and arrival information and to the beam point where it has been formed.	**GA_Driver.h**
Beam_pt *bp;	Pointer to Beam-Point.	
char ph_id[GA_PHASE_NAME]	Phase ID for *Driver* arrival.	
StaPt *stpt;	Pointer to station info for beam point.	
Phas_Inf *phspt;	Pointer to phase information for this *Driver*-phase-beam point.	
Sta_Ar *sta;	Pointer to station arrival structure for first arrival station.	
Arrival_Inf *ar;	Pointer to Arrival_Inf structure for first arrival station given *Driver.*	
Cor_Sta *csta;	Pointer to corroborating stations and arrivals list.	
double or_time;	Origin time for the *Driver.*	
double or_tmin;	Min. origin time for the *Driver.*	
double or_tmax;	Max. origin time for the *Driver.*	
double dr_mag;	*Driver* magnitude.	
double qfact;	Association quality factor. This is the combined probability for all corroborating phases to be associated with this *Driver.*	
double cr_ev_qual;	Event-based quality as measured by conflict resolution, Q_i.	
double res_norm;	Network-based probability of detection residual norm, measured as residual/sigma, used in event confirmation.	
double weight;	Current "weight" of *Driver* obtained by adding all weights for associated arrivals, used in event confirmation.	
int num_obs;	Total number of arrivals for this *Driver*, including *Driver.*	
Origin *origin;	Standard CSS DB 3.0 origin table.	
Origerr *origerr;	Standard CSS DB 3.0 origerr table.	
Assoc *assoc;	Standard CSS DB 3.0 assoc table (of length, num_obs).	
Assoc_CR *assoc_cr;	Association-based conflict resolution info.	
Driver *next;	Pointer to next *Driver* in linked list.	
}		
typedef struct assoc_cr Assoc_CR **{**	Contains association-based quantities used in resolving conflicts.	**GA_Driver.h**
Arrival_Inf *ar;	Arrival_Inf pointer.	
double chi2_fit;	Chi-squared fit level for arrival.	
double cr_ar_qual;	CR arrival quality measure.	
int rank;	Quality-based rank of given *Driver*/arrival pair. 0 is "keeper"; 1 is highest quality "loser"; etc.	
}		

5.0 Design for the Association of Secondary Phases and Late Arriving Data

There are several functions provided by **ESAL** in Phase 2 of the current *Global Association System* which must be implemented in Phase 3 to support the bulletin generation requirements of the ADSN, IDC and PNDC. In this section we describe the operational requirements and a design to meet them within the *Global Association System.* The requirements include:

- Association of non-defining phases
- Association of later phases that arrive after the initial time interval
- Processing of late-arriving data
  - Differentiated handling of data from primary and auxiliary networks
  - Efficient construction of new events from late-arriving data
  - Constrained refinement of previous event solutions with late-arriving data

Each of these tasks is described in the following section. Most of these can be handled by augmenting **GAassoc** and **GAconflict**. Refinement of previous event solutions by the association of phases that arrive after the initial time step, however, will require a new module, **GArefine**. A second new module, **GAmerge**, will be necessary to allow efficient processing of late-arriving data. The high-level processing and data flow for the complete system, including these two modules, are shown in Figure 11.

The main addition to the current system will be a set of shared libraries which will allow run-time prediction and association of phases to an existing event using its computed location and magnitude. These libraries will be used in **GAassoc** to add non-defining phases to located events that have passed conflict resolution and in **GArefine** to add late-arriving data to previously formed events. This functionality will be described in Section 5.1.

5.1 Association of Non-Defining Phases

GAassoc currently only associates phases that are required to define an event and compute its location (*Defining Phases*). It is important, however, to also identify and associate non-defining phases in order to reduce the number of false alarms that are constructed from these phases. These phases may also be important for subsequent event characterization.

It will be necessary to predict and associate non-defining phases in three different processing time frames:

1 - prompt arrivals within the same processing interval as the defining phases
2 - prompt arrivals within a later processing interval than the defining phases
3 - late arrivals

The term "prompt arrivals" is used to refer to data that are in the database at the time of the initial **GA** processing; "late arrivals" refer to data that were not in the database at the time of the initial **GA** processing and whose association requires reprocessing of the data. We focus on the first time frame here; differences in the processing of the second and third time frames will be discussed below.

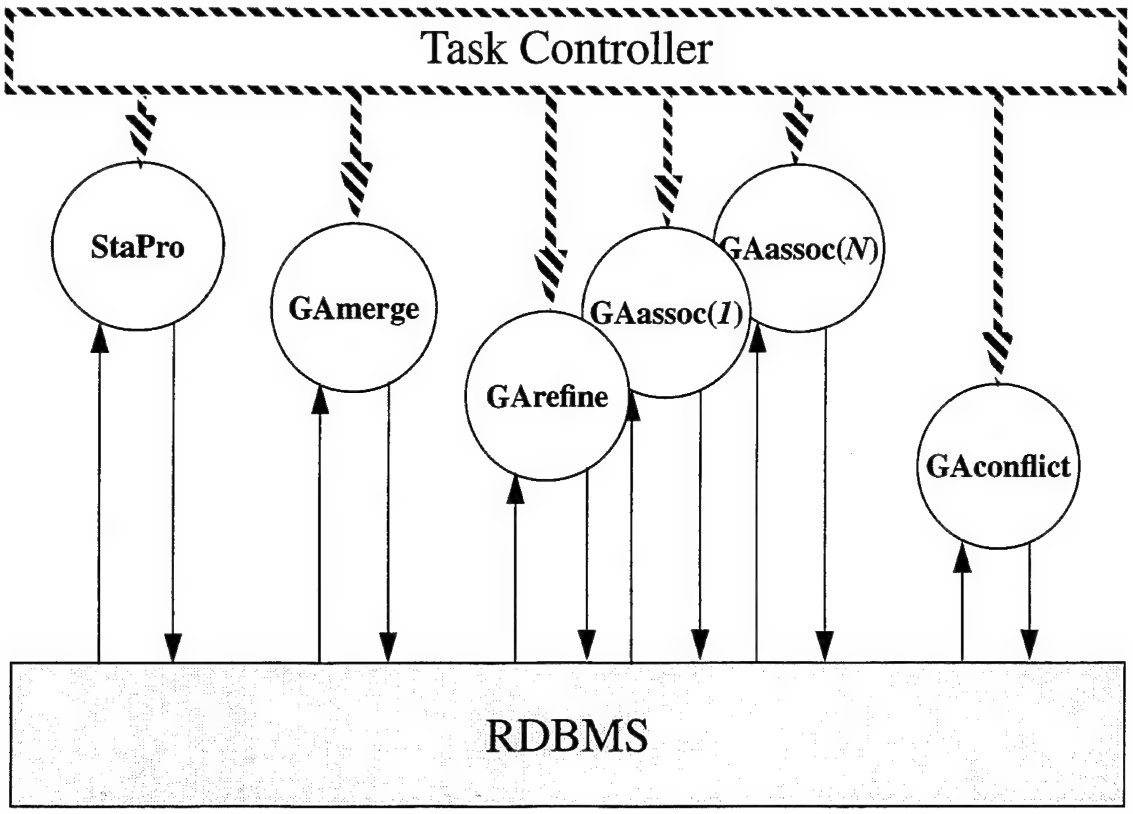

Figure 11. This figure shows the addition of **GAmerge** and **GArefine** and elimination of **EServer/ESAL** in the high-level processing and data flow for Phase 3 of the *Global Association System*. Process flow is indicated by dashed lines and data flow is indicated by solid lines. Note that **GArefine** can be run in parallel with the *N* instances of **GAassoc**.

There are several possible choices for when and how to predict and associate prompt non-defining phases in the same time interval as the defining phases. The simplest option would be to predict and associate them at the same time and in the same manner as the defining phases. However, this would be very inefficient. Non-defining phases do not contribute to, and are not required for, the formation and confirmation of an event. Since the number of preliminary events is dramatically reduced by location outlier, event confirmation, and conflict resolution tests, it is much more efficient to delay the prediction of non-defining phases until after these tests are applied. This efficiency gain is expected to outweigh the benefit of using precomputed values stored in the grid. In addition, the event magnitude can be used to restrict which *phaseids*[1] are predicted, and this will obviously be more accurate after the final event location is computed.

Another option would be to add the non-defining phases in a separate process after all defining phases for a time step have been processed and merged. This is the model used in Phase 2 of the *Global Association System*. It is efficient and modular, but it reduces the level of parallelization that

1. We use the term *phaseid* to refer to a phase identifier, e.g. P. A *phase* is an arrival with an assigned *phaseid*.

is made possible by running multiple **GAassoc** processes on separate sectors. Instead, we prefer a model that predicts and associates non-defining phases within each sector after event formation and intra-sector conflict resolution are complete, but before the results from each sector are merged. Conflicts generated by associating non-defining phases will then be resolved in a second pass through conflict resolution within **GAassoc**. This will require adding two steps to **GAassoc** after conflict resolution: (1) Predict and associate non-defining phases, and (2) Resolve conflicts generated by these associations (compare Figure 12 to Figure 6). These new steps are described in the following two subsections.

5.1.1 Prediction and Association of Phases

Prediction and association of phases to previously-located events will be required for association of prompt non-defining phases (in the same time interval as the defining phases or in a later one) and late-arriving data. Consequently, this functionality will be implemented as a set of shared libraries which will be used by **GAassoc** and **GArefine**.

Restrictions on the Prediction and Association of Phases

Amplitude tables are generally not available for secondary phases, so amplitude consistency tests cannot always be applied. Instead, a set of related heuristic restrictions will be imposed as is done in **ESAL**. First, the distance range at which *phaseids* can be associated, and the magnitude of the event to which they can be associated, will be restricted. These restrictions are expressed as:

> *Phaseid, PID, will be predicted for event, EV, at station, STA,*
> *if and only if the magnitude (mb) for EV is greater than or equal to minmag(PID)*
> *and the distance from EV to STA is between mindist(PID) and maxdist(PID)*

It may eventually be desirable to store these restrictions in the grid, but the restrictions currently used in **ESAL** are neither source nor path-specific. It will therefore be more efficient to store them in a separate structure accessible by *phaseid*, magnitude and distance. These data will be read from a UNIX file that can be merged with the *dist_depth_range_file* used by **GAcons**.

The prediction of non-defining phases will also be restricted according to the setting of the current user-parameter, *primary_required_for_secondary.* When this restriction is set, non-defining phases will only be predicted for stations that have an associated defining phase. In addition, user-parameters will control restrictions such as: a secondary phase may not be associated unless azimuth and slowness data are available (e.g. *secondary_requires_slow_az*), and a later phase which has been grouped by **StaPro** may not be associated with an event unless the earliest phase in the group has also been associated (e.g. *earliest_stapro_phase_required*).

Prediction of Phaseids

Predictions will be based on the computed event location and error ellipse. This will be done using a prediction routine like the one used by **ESAL** and standard travel time tables. The routine used by **ESAL** is similar to the one used by **GAcons** to calculate travel times for the grid, but it uses the error ellipse size and orientation to calculate origin uncertainties. These routines do not currently include the depth uncertainty for fixed-depth locations, but this can be addressed initially by modelling the depth uncertainty as a linear function of the depth (e.g. with user parameters *prediction_depth_model_constant* and *prediction_depth_model_slope*). For a given event, theoretical travel times, azimuth, slowness, and their uncertainties will be predicted for the non-

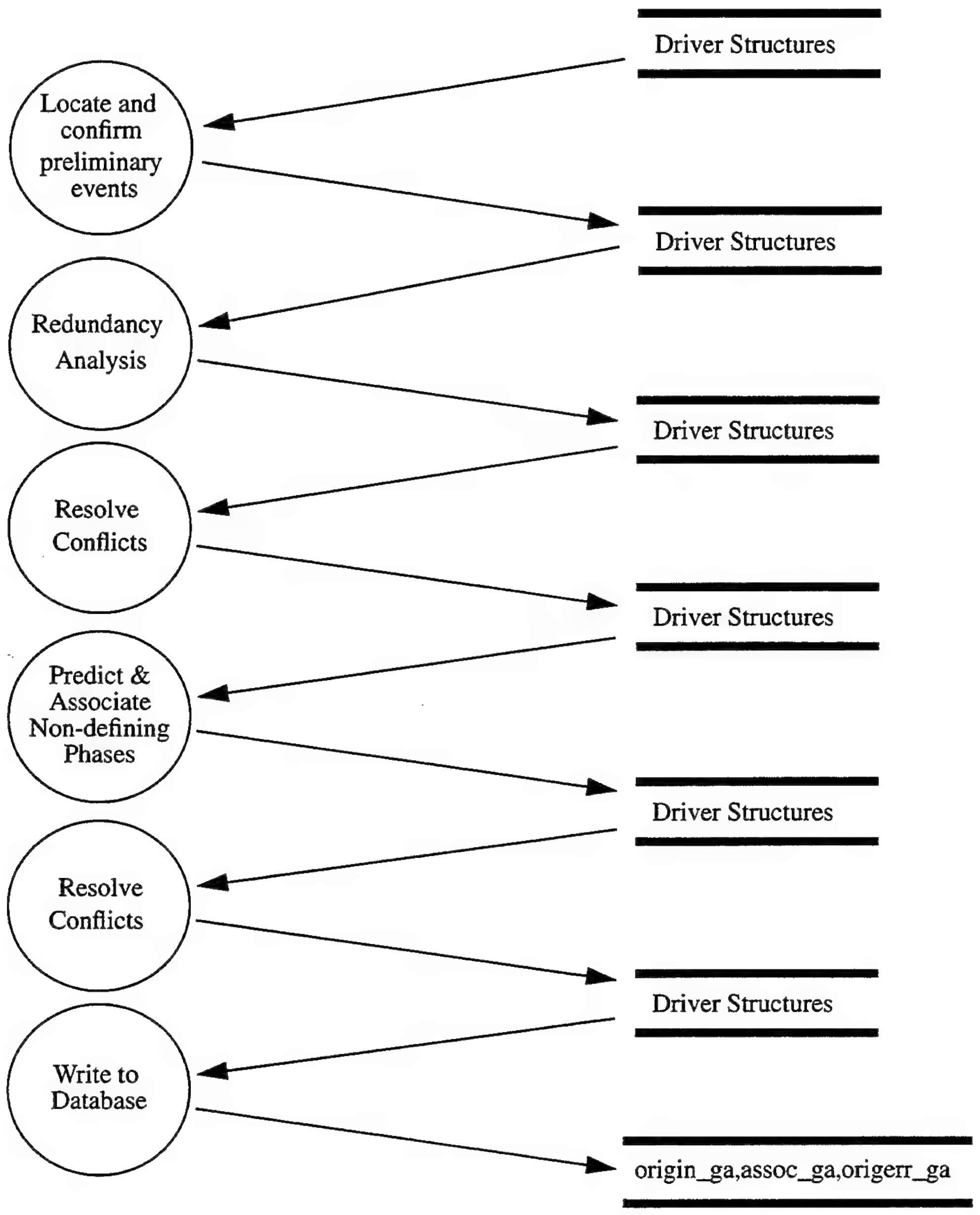

Figure 12. Level 3 DFD diagram for **GAassoc,** starting with location. Prediction and association of non-defining phases and a second pass through conflict resolution have been inserted after the initial pass through conflict resolution.

defining *phaseids* (specified by a user parameter, e.g. *nondefining_phases*) that satisfy the restrictions described above.

For a given event, station and *phaseid*, residuals will be computed for every phase which is not already associated with the event, has an arrival time within a time window of the theoretical travel time, and has an initial *phaseid* that matches or can be changed to the predicted *phaseid* (*forward_transformation_list*). Association will be based on a chi-square test of the location residuals (time, azimuth and slowness) normalized by the RMS sum of the data standard deviation (e.g. deltim) and the origin uncertainty computed for the error ellipse. The chi-square sum will be compared to the user parameter *chi_outlier* used in the location outlier test. The time window for prediction will be based on the maximum time residual that could pass the chi-square test.

Association of Phases

It is possible that more than one arrival will fit the predicted values for a single *phaseid*, or that one arrival will fit more than one *phaseid*. To resolve this ambiguity, we will collect and order all possible associations for a station (e.g. in tuples like (arrival, *phaseid*, fit)). The phase with the best chi-square fit will be accepted and all other preliminary associations for the same arrival or *phaseid* will be removed from the list. This will continue until all ambiguities are resolved. The remaining phases will be associated with the event.

5.1.2 Conflict Resolution

Association of non-defining phases will not change event parameters so it will not be necessary to relocate or reapply event confirmation criteria after association. However, conflicts may be generated between the newly associated phases and other non-defining or defining phases. It will, therefore, be necessary to make a second pass through conflict resolution to identify and resolve such conflicts, as shown in Figure 12. If a non-defining phase is removed due to conflict resolution, nothing more must be done. If a defining phase is removed, however, redundancy and event confirmation tests will need to be reapplied and surviving events must be relocated.

The conflict resolution algorithm described in Section 4.2 will need to be modified slightly in order to work correctly with non-defining phases:

- The χ^2 term in F_{ij} will need to be modified to incorporate the origin uncertainty characterized by the error ellipse in the scaling of the residuals.
- The amplitude component of the χ^2 term in F_{ij} will generally not be available.
- The Q_2 term, characterizing the likelihood that the event will be dissolved if the phase is removed, will need to be modified to be dependent on whether the phase in question is defining or not.
- If a removed phase is non-defining, relocation of the event and reevaluation of the conflict-resolution quality measures is not required.

More experience with the conflict resolution algorithm will be necessary to determine if the following changes are desirable:

- Include non-defining phases in the initial clustering algorithm.
- Add a term to Q_1 which allows non-defining associations to contribute to the confidence that an event is real.

5.2 Association of Later Phases

GAassoc processes arrivals in a series of overlapping time intervals and only considers arrivals within a single interval[1]. Currently **ESAL** handles the association of any phases that arrive after the end of the time interval in which an event is formed. To include this functionality in the *Global Association System* we will introduce a new module called **GArefine**. This module will share the code used in **GAassoc** to predict theoretical travel times for non-defining phases.

The purpose of **GArefine** is to refine and augment previously confirmed events, not to significantly revise them or to generate new events; all new events will still be constructed by **GAassoc**. Data will be read from and written to the *Final_tables*, which are the output of **GAconflict**. Phases in the current time step which might be associated with a previous event will be predicted and associated if appropriate. Internal conflicts will be resolved within **GArefine**, while any conflicts generated with the results of **GAassoc** processing of the current time step will be resolved during subsequent processing by **GAconflict**. During processing of prompt data, the main concern is association of non-defining phases since the size of the time interval is generally selected to allow inclusion of all defining phases. However, the approach we select must be flexible enough to allow the association of defining phases as well.

The general processing flow of **GArefine** is shown in Figure 13. The first step of **GArefine** will be to query the *Final_tables* to determine which previous events could predict a missing phase in the current time step. As described above, prediction of each *phaseid* will be restricted to a specified distance/time range and minimum event magnitude. Prediction will also be restricted to those *phaseids* for which there is not already an association. Only those previous events which, based on their magnitude and origin time, could predict a missing phase in the current time step will be loaded into **GArefine**. They will be loaded into standard **GAassoc** *Driver* structures using the utilities developed in **GAconflict** to load previous events for conflict resolution. All arrivals associated with these events will also be loaded, plus all arrivals in the current time step. Note that arrivals for the lookback before the current time step will not be loaded since they have been fully processed in an earlier time step. Consequently there will be no overlap between the new arrivals and those already associated. If there are no eligible previous events then **GArefine** will exit.

Defining *phaseids* (specified by *phases*) will be predicted and associated first. As in **GAassoc**, which missing *phaseids* should be predicted for each station will be determined based on the event magnitude, origin time, and association set. The theoretical travel times, azimuth, slowness,

1. A time interval is defined by three parameters: the *start_time*, the *end_time*, and the *lookback*. We define the "current time step" as the period from *start_time* to *end_time*; the "current time interval" is the current time step plus the lookback, which precedes the current time step.

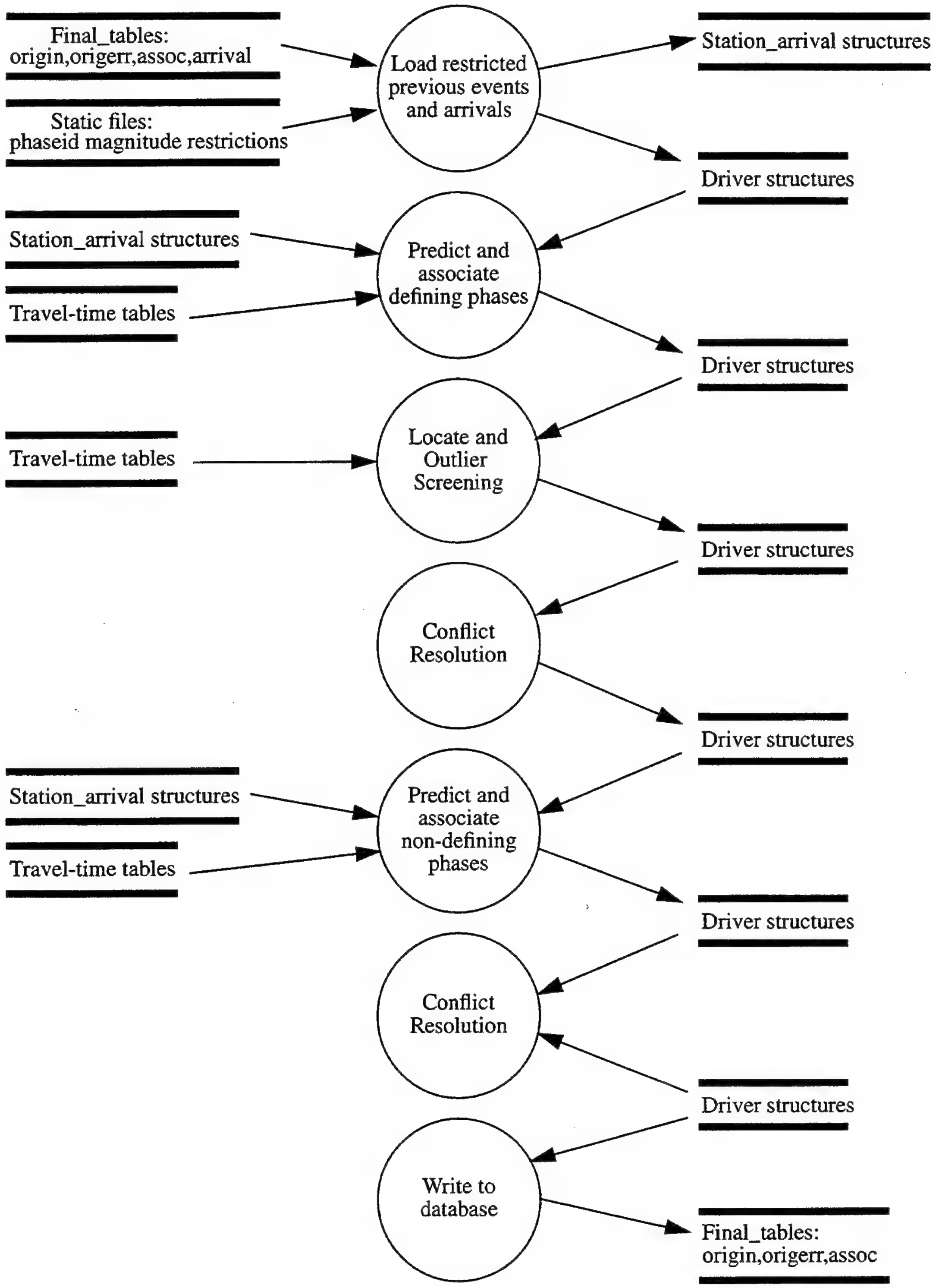
Final_tables: origin,origerr,assoc,arrival
Static files: phaseid magnitude restrictions
Load restricted previous events and arrivals
Station_arrival structures
Driver structures
Station_arrival structures
Travel-time tables
Predict and associate defining phases
Driver structures
Travel-time tables
Locate and Outlier Screening
Driver structures
Conflict Resolution
Driver structures
Station_arrival structures
Travel-time tables
Predict and associate non-defining phases
Driver structures
Conflict Resolution
Driver structures
Write to database
Final_tables: origin,origerr,assoc

Figure 13. DFD for **GArefine**. **GArefine** will share subroutines with **GAconflict** to load data from the database and with **GAassoc** to predict and associate phases, for location, outlier screening, and conflict resolution, and to write to the database.

and their uncertainties will be predicted for those *phaseids* using the event location and error ellipse and residuals for all unassociated phases will be computed within a time window of the theoretical travel time which has an initial *phaseid* that matches or can be changed to the predicted *phaseid*. Note that phases which are already associated will not be predicted since that would be repeating previous work. Conflicts between potential associations will be resolved as in **GAassoc**.

All defining phases which pass the association tests will be associated at once, followed by relocation and outlier screening as performed in **GAassoc**, with two differences:

- If one of the original defining phases is the worst outlier, the worst newly-associated phase will be removed instead.
- If the locator fails (e.g. fails to converge) during this process, the original event will be restored.

These differences are motivated by our intention to refine and not revise the original events. An alternative to this sequence would be to associate new phases one at a time, followed by relocation and reprediction. This might be a more stable sequence for refining a small, poorly constrained event, but it would in general be more computationally expensive since it would require a relocation and reprediction for each associated phase.

After all events have been relocated and screened for outliers, conflicts will be identified and resolved as in **GAassoc**. Since only new arrivals are predicted, the only conflicts will be between different associations of the new arrivals. As in outlier screening, conflict resolution will be restricted so it does not degrade the original event.

Once all defining phases have been associated and conflicts among them resolved, non-defining phases will be predicted and associated, followed by a second pass through conflict resolution. This will proceed exactly as it did in the association of non-defining phases within **GAassoc**.

Results will be written back to the *Final_tables*. It will be necessary to delete the previous events from these tables before the revised ones are written out.

After **GAassoc** is complete for all sectors and **GArefine** is complete for previous events, **GAconflict** will be run to resolve any remaining inter-sector and inter-time-step conflicts, as shown in Figure 11. The algorithm described in Section 4.2 to identify conflicts with previous events will need to be extended to include conflicts with non-defining phases. Otherwise, the conflict resolution algorithm will be the same as described above for a single time step when non-defining associations are included.

5.3 Processing of Late-Arriving Data

Processing of late-arriving data implies reprocessing an interval which contains data that was not present when the data was first processed. A couple of distinct scenarios must be supported to satisfy the operational requirements of the ADSN and IDC. In the ADSN, the initial processing has generally been reviewed and corrected by an analyst. These results must be preserved when adding late data. In one scenario, late arrivals may be associated with previously-formed events (Previous Events) but the original set of associations and event parameters (location and magnitude) may not be changed and no new events are to be formed. In a second scenario, late arrivals may be associated and event parameters recalculated, but the original set of associations may not be changed. New events may be formed as long as they do not involve previously-associated arrivals. In the IDC, the Previous Events have not been reviewed by an analyst, but the late data consists primarily of data from an auxiliary network that has been requested to refine the Previous Event locations. In that case, the primary data will take precedence over the auxiliary data to ensure the effect is refinement and not revision. Furthermore, because of the large volumes of data involved, this process should be as efficient as possible. Obviously, it would be very inefficient to completely reprocess the interval to allow association of a small number of late arrivals.

The processing flow will follow the model used in the processing of prompt data, with addition of a preliminary step to merge previous results into the current processing (Figure 11). As in the processing of prompt data, **GArefine** will be used to augment previous event solutions and **GAassoc** will be used to generate new ones.

The first step in late processing will be to process the late arrivals through **StaPro** to obtain *phaseids*, as it is in the current hybrid system. The **Process Manager** [*Given et al.*, 1993] keeps track of intervals of data at each station which have not been processed by **StaPro**. Only these intervals will be processed during late processing.

The next step will be to merge the results of an earlier run into the current database. The previous results for a given time step must be merged before either **GAassoc** or **GArefine** are run to make them available to constrain the processing. Although this function could be combined with **GArefine**, separating it into a separate process which will be run before **GArefine** and **GAassoc** for a given time step will allow **GArefine** to be run in parallel with **GAassoc**; we will call this module **GAmerge**. We assume a database model where **GAassoc** writes its results to one set of database tables, called the *GA_tables,* and **GArefine** and **GAconflict** write their results to a separate set of tables, called the *Final_tables*. We further assume that the results of the previous run are in a third set of tables, called the *Previous_tables,* and that there is a single self-consistent **arrival** table which contains all of the arrivals for these three sets of tables. **GAmerge** will copy **origin**, **assoc**, and related records from the *Previous_tables* to the *Final_tables* when the first association of the event lies in the current time step (i.e., the arrival time is between *start_time* and *end_time).* This database model is shown in Figure 14.

5.3.1 Differentiated Handling of Data from Primary and Auxiliary Network

The IDC utilizes the concept of a primary and auxiliary network of stations. Data from the primary network are used to define an event while data from the auxiliary network are used only to refine the event location. It will therefore be necessary to support the following constraints:

- *Drivers* may only be formed from arrivals at stations in the primary network.
- Only arrivals at stations in the primary network contribute to event confirmation tests.

The first constraint could be implemented in **GAcons** by restricting the stations that are considered when determining First-arrival stations [*Le Bras et al.*, 1994a]. This, however, would require rebuilding the grid if a station is moved from one network to the other. A more flexible approach, which could be used for both constraints, would be to read the stations' network membership from the **affiliation** database table during initialization of **GAassoc**. A user parameter, e.g. *primary_network*, would identify the primary network, and all arrivals at stations not in that network would be labelled as auxiliary arrivals. This would allow an efficient run-time restriction on

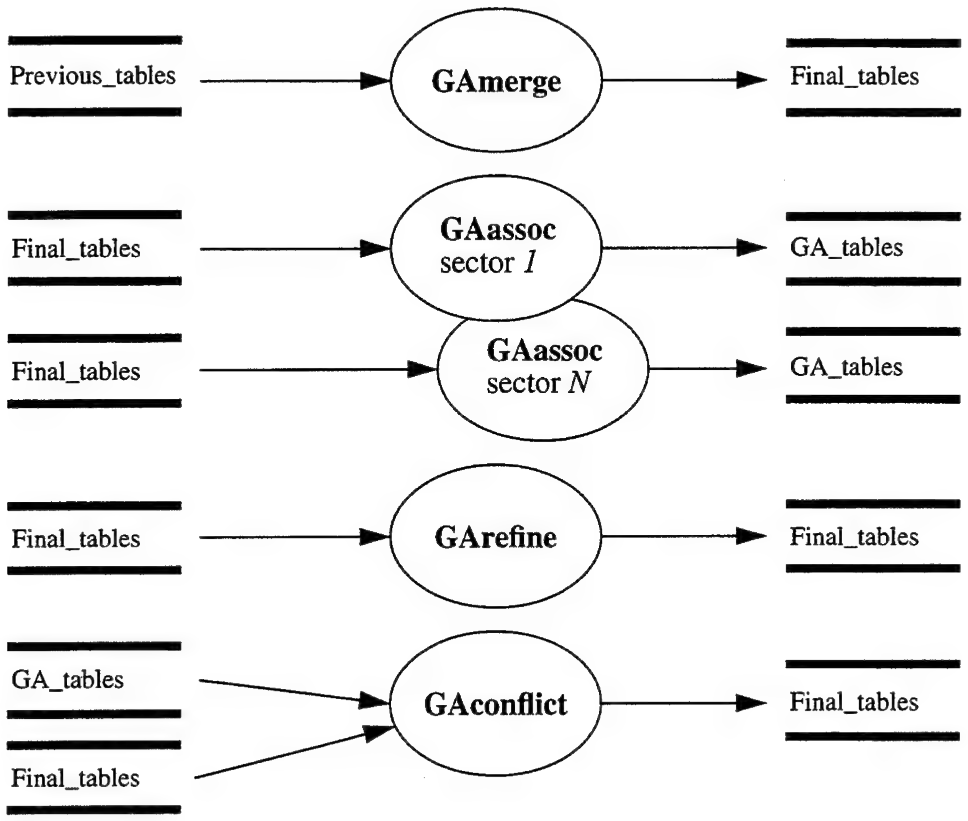

Figure 14. This figure shows the different sets of event tables read and written by the different **GA** processes. There is a single arrival table read by **GAassoc, GArefine** and **GAconflict**. The *Final_tables* are read by **GAassoc** for constraint purposes only.

the generation of *Drivers* and a common mechanism to be used to restrict the calculation of the Weighted-Count and Network-Probdet event confirmation tests [*Le Bras et al.*, 1994a]. If a *primary_network* is not specified then the restriction will not be applied.

5.3.2 Efficient Construction of New Events from Late-Arriving Data

It is, of course, possible to produce events incorporating late-arriving data by simply reprocessing all of the data, including the late arrivals, through **GAassoc**. We would like, however, to improve the quality and efficiency of the processing by taking advantage of the earlier processing and analysis. To meet the operational requirements of the ADSN and IDC for the processing of late-arriving data described earlier we must support two specific cases. In one case we will lock associations that have been reviewed by an analyst so they cannot be changed, for example, by being associated with another event. In the second case we will restrict event generation to require the use of late-arriving data. This will avoid the wasted effort of constructing and reevaluating preliminary events that have already been considered. Both of these cases can be accomplished by modifying the **GAassoc** module.

The first case will be implemented by simply restricting the arrivals that are loaded into **GAassoc** to not include any that are associated to an analyst-reviewed event. This, of course, requires identifying analyst-reviewed events. One way to do this would be to use the **origin** auth field. If **GAassoc** writes a distinct auth field (e.g. starting with *ga_auth_prefix*) when it forms new origins, then an **origin** auth field that was not written by **GAassoc** would indicate that the origin had been reviewed by an analyst. This constraint will be controlled by the user parameter *lock_analyst_associations.* If it is not applied, all arrivals will be loaded. No other change will be required for this case.

In the second case we wish to avoid wasting time reconsidering preliminary events that were already evaluated during prior processing. We will do this by requiring preliminary events to associate late arrivals. There are two ways to accomplish this. One is to form preliminary events in **GAassoc** as normal, but to immediately abandon them if they do not associate a late arrival. This would be significantly more efficient than completely reprocessing the data, but not as efficient as a more restrictive approach that restricts the *Driver* used in **GAassoc** to be a late arrival; this is the approach we recommend. This behavior will be controlled by a user parameter, e.g. *restrict_drivers_to_late_arrivals.*

Late arrivals will be identified by using the author field of the **arrival** table. Late-arriving data can be thought of as data which has been processed by **StaPro** but which has not been fully processed by **GAassoc. StaPro** currently updates the author field of arrivals it has processed. If we modify **GAassoc** to update the author field of an arrival at the completion of its final pass through **GAassoc**[1], we can then identify a late arrival as one where auth equals *stapro_auth.*

GAassoc will require a couple of modifications to allow it to make full use of a late arrival as a *Driver* because it normally assumes the *Driver* is the first arrival from the event. **GAassoc** restricts a *Driver* to be an arrival at a station identified in the grid as a First-arrival station [*Le Bras et al.* 1994a] and to have a detection time between *start_time* and *end_time.* In addition, only phases arriving after the *Driver* are considered as possible corroborating phases. These restrictions will have to be relaxed and all phases be considered since the *Driver* may be any defining phase in this case. These modifications will only apply when *Drivers* are restricted to be late arrivals.

When **GAassoc** is restricted in its use of arrivals for *Drivers*, either to late or primary data, it will check the database for eligible arrivals before any other processing or loading of data and will terminate if none are available.

5.3.3 Constrained Refinement of Previous Events with Late-Arriving Data

Event refinement will be handled by **GArefine** and will progress basically as described for prompt processing and shown in Figure 13, with the identification and loading of events that might predict a new phase in the current time step. An important difference, however, will be that only late, rather than all, arrivals in the current time step will be loaded. If there are no late arriv-

1. An arrival will be processed by **GAassoc** two or more times, depending on the ratio of the *lookback* to the time step duration. The final pass can be identified by having the arrival time between *start_time* and *end_time*

als in the current time step then **GArefine** will exit. The events that are loaded will be checked to determine if they have been reviewed by an analyst and will be marked accordingly; associated arrivals will be marked similarly. Analyst-reviewed events and late, primary and auxiliary data will be identified as described earlier.

As stated above, certain constraints may be applied to preserve the association sets and locations of analyst-reviewed events. There may also be the constraint that the association of an auxiliary arrival may not cause the disassociation of a primary arrival; these constraints will be controlled by a set of user parameters, e.g. *lock_analyst_associations*, *lock_analyst_locations*, and *prefer_primary_associations*. These constraints will require the following modifications to **GArefine**:

- When analyst locations are locked, no location or outlier screening will be done. Output to the database will be modified so that input events will not be purged from the database but will be updated instead. Specifically, new associations will be added to **assoc** and the number of associated phases will be updated in **origin** if appropriate.
- When analyst associations are locked but the location is not locked, analyst-associated arrivals may not be removed by location outlier screening. If an analyst-associated arrival is an outlier and there is a **GA**-associated arrival, the latter will be removed instead. If an error occurs during location (e.g. failure to converge) then the original analyst-reviewed event will be restored.
- When primary associations are preferred, if a primary association is a location outlier and there is also an auxiliary association, the latter will be removed instead.

The rest of **GArefine** will proceed as when processing prompt data, with one small change. Although the original associations of an analyst-reviewed event cannot be in conflict, it will be possible for a conflict to arise when a later phase is associated with both an analyst-reviewed and an **GA**-produced event. We will therefore want to increase the event quality component of the conflict resolution event quality measure, Q_1, for analyst-reviewed events, probably to unity.

References

Anderson, J., M. Mortell, B. MacRitchie, and H. Turner, Generic Database Interface (GDI) User Manual, *Tech. Rep. SAIC-93/1001 REV,* Science Applications International Corporation, 1994.

Bratt, S., G. Beall, H. Swanger, F. Dashiell and T. Bache, A knowledge-based system for automatic interpretation of seismic data to associate signals and locate events, *Tech. Rep. SAIC-91/1281*, Science Applications International Corporation, 73 pp., 1991.

Bratt, S., G. Beall, H. Swanger, F. Dashiell and T. Bache, A knowledge-based system for automatic interpretation of seismic data to associate signals and locate events, *Tech. Rep.* [in progress, revised edition of *SAIC-91/1281*], Science Applications International Corporation, 1994.

Given, J., W. Fox, J. Wang and T. Bache, The Intelligent Monitoring System: Software Integration Platform, *Tech. Rep. SAIC-93/1069*, Science Applications International Corporation, 32 pp., 1993.

Kerr, A. (ed.), Overview GSETT-3, Report prepared by the *GSE Working Group on Planning*, 9 pp., October, 1993.

Le Bras, R., H. Swanger, T. Sereno, G. Beall, R. Jenkins and W. Nagy, Global Association. Design Document and User's manual, *Tech. Rep. SAIC-94/1142*, 67 pp., 1994a.

Le Bras, R., H. Swanger, T. Sereno, G. Beall, R. Jenkins, W. Nagy and A. Henson, Global Association Final Report, *Tech. Rep. SAIC-94/1155*, 28 pp., 1994b.

Leonard, S., Automatic global event association and location estimation using a knowledge based approach to generalized beamforming, Proceedings of the 15th Annual PL/ARPA Seismic Research Symposium, *PL-TR-93-2160*, 248-255, 1993.

Ringdal, F. and T. Kværna, A multi-channel processing approach to real time network detection, phase association, and threshold monitoring, *Bull. Seismol. Soc. Am.*, *79*, 780-798, 1989.

Taylor, D. and S. Leonard, Generalized beamforming for automatic association, Proceedings of the 14th Annual PL/ARPA Seismic Research Symposium, *PL-TR-92-2210*, 422-428, 1992.

Appendix A: GAassoc and GAconflict Parameter Descriptions

This appendix contains descriptions of the user-parameters for **GAassoc** and **GAconflict.** The user-input to both programs is through command-line arguments. These arguments can be stored in a parameter file, and the name of the file is specified on the program command-line (e.g. *GAconflict_par=GAconflict.par*).

GAassoc and **GAconflict** share a number of modules and, therefore, a number of parameters are valid for both applications. The following list of parameters is divided between **GAassoc** parameters (A.1), **GAconflict** parameters(A.2) and shared parameters (A.3). Shared parameters are valid for both applications. Refer to *Le Bras et al.* [1994a] for description of user-parameters for **GAcons**. Also provided in this appendix are sample parameter files for **GAassoc** and **GAconflict** (A.4).

A.1 GAassoc User-Parameters

A.1.1 Database Interface and Input Files Parameters

gdihome:
Root directory location of GDI (Generic Database Interface) installation. Default is set based on the GDI_HOME environment variable.

vendor:
Name of the database vendor (e.g., "oracle"). Default is NULL.

server:
Name of the database server. Default is NULL.

constr:
Type of database tuple constructor. Default is NULL.

database:
Name of the device where the database is located (e.g. "t:skrymir:dev6037"). Defaults to the setting defined in the user environment (e.g. TWO_TASK environment variable). Required.

account:
Database account name. Required.

maxrecs:
Maximum number of records to read from the database. If -1, all records will be returned. Default is 30000.

in-arrival-table:
Name of the input **arrival** table. The detections to be associated are read from this table. Station processing should be run on the detections before using **GAassoc**. Required.

in-assoc-table:
Name of the input **assoc** table. The belief field from this table is read for each arrival. It is also used to screen arrivals associated with local events with *ML* magnitude less than a user-specified

value. Required.

in-origin-table:
Name of the input **origin** table. This is used to screen local events with *ML* magnitude less than a user-specified value. Required.

in-origerr-table:
Name of the input **origerr** table associated with the input **origin** table. Required.

out-origin-table:
Name of the output **origin** table. This will contain the **origin** records for the preliminary events formed by **GAassoc**. Required.

out-assoc-table:
Name of the output **assoc** table. This table contains the **assoc** records associated with the preliminary events formed by **GAassoc**. Required.

out-origerr-table:
Name of the output **origerr** table. This table contains the **origerr** records associated with the preliminary events formed by **GAassoc**. Required.

net:
Name of seismic network (e.g. GSETT3). Required.

minimum_ml_previously_determined:
Minimum local magnitude of events whose arrivals will be tentatively associated by **GAassoc**. Arrivals that have been associated by station processing with local events of magnitude less than this value are not considered for association. Default is 2.0.

input_path:
Path name to the directory where the input grid file produced by **GAcons** is located. Required.

input_file:
Name of input grid file produced by **GAcons**. This file contains the grid information to be used by **GAassoc** in the forming of preliminary events. This file name must be specified.

table_path:
Path name to travel time and magnitude tables directory. Required.

atten_file:
Path name and file name for the attenuation tables used for probability of detection calculations. Required.

mb_dist_depth_suffix:
Suffix for *mb* tables. Required.

A.1.2 Association Loop Parameters

start_time:
Epoch time of the starting time for analysis.

end_time:
Epoch time of the end time for analysis.

lookback:
Time in seconds before the start time to include in the analysis. Events with origin time between *start_time-lookback* and *end_time-lookback* are considered in the analysis. Arrivals between *start_time-lookback* and *end_time* are read and considered in the analysis.

belief_threshold:
Threshold value for the belief field in the **arrival** table. If the belief (assigned by **StaPro**) is above this threshold the phase identification cannot be changed by **GAassoc**.

phases:
Names of phases to be used in the association. This must be a subset of the phases used in **GAcons**. Required.

primary_phases:
Names of phases to be considered "primary" phases. Required.

num_first_sta:
Maximum number of "first-arrival stations" to use from the grid file produced by **GAcons**. The stations are ordered with the highest probability station first. Default is 10.

count_limit:
When the number of preliminary events formed in the association loop of **GAassoc** reaches this limit, redundancy analysis, event splitting and event confirmation are performed before more preliminary events are formed. This is an efficiency measure meant to recycle memory space before a large portion of the physical memory is used-up. When memory and swap space are limited, set this limit to a small number (e.g. 10000). Set this number to a very large number (e.g. 1000000) to prevent the recycling from happening when sufficient space is available on your hardware. Default is 10000.

freeze_arrivals_at_beam_points:
If this string is present in the parameter file, arrivals will be frozen at each beam point once associated with a preliminary event. If this string is not present in the parameter file no freezing is performed. The consequences of freezing the arrivals are that less preliminary event will be formed and there is a chance (albeit small) that valid events will be missed. The preliminary events list will be more exhaustive when no freezing is performed. Default is FALSE.

primary_required_for_secondary:
If this string is present in the parameter file, an arrival can be associated to a preliminary event only when there is a corresponding primary phase from the same station already associated to that event. Default is FALSE.

regional_S_phases:
List of regional *S* phases. A regional *P* phase cannot be associated with a grid cell at teleseismic distance if station processing grouped it with a compatible *S* phase in the list of *regional_S_phases*. Required.

forward_transformation_list:
For each phase identified by station processing, this list restricts the phase identifiers (*phaseids*) that it can be transformed into by **GAassoc**. The format for this parameter is a list of lists as in "(P PKPdf Pn), (S Sn Lg Rg)". The parenthesis-enclosed list contains the station-processing assigned *phaseid* as its first member, followed by the *phaseids* that this *phaseid* can be transformed into by **GAassoc**. If a *phaseid* is not present in that list, **GAassoc** is not allowed to modify the phase identification of a particular arrival into that *phaseid*. Phase identifications will be carried over from station processing if no specific identification is made (e.g Tx not identified). Required.

sigma_time:
Sigma factor for time measurement uncertainty. This factor is multiplied by the deltim standard deviation to determine the interval for the preliminary screening done during the search for corroborating phases. Default is 3.0.

sigma_slowness:
Sigma factor for slowness measurement uncertainty (see the *sigma_time* description). Default is 3.0.

chi_limit:
Threshold value for the chi-square association test. This is used within the association loop to determine if a corroborating arrival belongs to a preliminary event formed by a *Driver* arrival. The default is 0.99, meaning that there is a 1 percent chance that the arrival belongs to the same event as the *Driver* arrival, this event being located within the grid cell being examined.

A.1.3 Optional Processes Parameters

save_previous_tables:
A boolean variable to control overwriting of existing database tables. If TRUE, previously computed database records will be saved; else, they will be removed. Under normal operating conditions this should be set to FALSE (string should be omitted from parameter file). Default is FALSE.

redundancy_required:
If this string is present in the parameter file, a complete redundancy test is performed after the association loop. This is done in the normal operating mode. Default is FALSE.

probdet_before_location:
If this string is present in the parameter file, a network probability test is performed prior to location. Parameters for this test are defined by *residual_over_sigma_max* and *max_obs_net_prob.* Default is FALSE.

probdet_after_location:
If this string is present in the parameter file, a network probability test is performed after location. Parameters for this test are defined by *residual_over_sigma_max* and *max_obs_net_prob.*Default is FALSE.

do_clustering:
If this string is present in the parameter file, clustering analysis will be performed. Default is FALSE.

do_association_based_conflict_resolution:
If this string is present in the parameter file, association-based conflict resolution analysis will be performed. Default is FALSE.

A.1.4 Verbose Level Parameters (GAassoc)

global_verbose:
Level of verbosity for overall **GAassoc** analysis. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). A level of 1 is recommended for general run-time purposes. The highest level, 4, is only intended for debugging. Default is 1.

assoc_verbose:
Level of verbosity for main association processing. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). Default is 0.

ev_verbose:
Level of verbosity for event confirmation screening. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). Default is 0.

cr_verbose:
Level of verbosity for conflict resolution analysis. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). A level of 2 is particularly useful when trying to understand why an event was removed due to conflict resolution. The highest level, 4, is only intended for debugging. Default is 0.

loc_verbose:
Level of verbosity for printed locator output. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). One will usually not want to look at specific locator output unless the user has some particular interest in the behavior of a single event location. Default is 0.

A.2 GAconflict parameters

A.2.1 Database Interface and Input Files Parameters

gdihome:
Root directory location of GDI (Generic Database Interface) installation. Default is set based on the GDI_HOME environment variable.

vendor:
Name of the database vendor (e.g., "oracle"). Default is NULL.

server:
Name of the database server. Default is NULL.

constr:

Type of database tuple constructor. Default is NULL.
database:
Name of the device where the database is located (e.g. "t:skrymir:dev6037"). Defaults to the setting defined in the user environment (e.g. TWO_TASK environment variable). Required.

account:
Database account name for input arrival table and output **assoc**, **origin** and **origerr** tables. Required.

maxrecs:
Maximum number of records to fetch from the database. If -1, all records will be returned. Default is 30000.

in-arrival-table:
Name of the input database table where the input arrival tuples are located. This table (**arrival**) contains the necessary raw event phase data information required to do conflict resolution and event location. Required.

in-assoc-table:
Name of the input database table where the input assoc tuples are located. Required.

in-origin-table:
Name of the input database table where the input origin tuples are located. Required.

in-origerr-table:
Name of the input database table where the input origerr tuples are located. Required.

out-origin-table:
Name of the database table where the output origin tuples will be written upon successful processing of events in **GAconflict**. Required.

out-assoc-table:
Name of the database table where the output assoc tuples will be written upon successful processing of events in **GAconflict**. Required.

out-origerr-table:
Name of the database table where the output origerr tuples will be written upon successful processing of events in **GAconflict**. Required.

net:
Name of unique seismic network identifier (e.g., GSETT3). Required.

table_path:
Directory and prefix for the travel-time and magnitude distance/depth correction tables. Required.

atten_file:
Directory and file name for the location of the attenuation tables used for probability of detection calculations. Required.

mb_dist_depth_suffix:

File name suffix for Mb distance/depth dependent magnitude corrections. Required.
start_time:
Epoch time at which to gather arrivals to begin analysis. Required.

end_time:
Epoch time at which to end analysis. Required.

phases:
List of acceptable *phaseids* to be used in the main association loop (e.g., "P,PKPdf,S,Pn,Lg,PcP"). This must be a subset of the phases used in **GAcons**. Required.

primary_phases:
List of acceptable "primary" *phaseids* to be used in the event confirmation process (e.g., "P,PKPdf,Pn"). Primary phases are given a weighting within event confirmation set by primary_time_weight. This list must be a subset of the list specified by *phases.* Required.

A.2.2 Optional Processes Parameters

redundancy_required:
If this string is present in the parameter file, a complete redundancy test is performed after the association loop. This is done in the normal operating mode. Default is FALSE.

do_clustering:
If this string is present in the parameter file, clustering analysis will be performed. Default is FALSE.

do_association_based_conflict_resolution:
If this string is present in the parameter file, association-based conflict resolution analysis will be performed. Default is FALSE.

A.2.3 Verbose Level Parameters (GAconflict)

global_verbose:
Level of verbosity for overall **GAconflict** analysis. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). A level of 1 is recommended for general run-time purposes. The highest level, 4, is only intended for debugging. Default is 1.

ev_verbose:
Level of verbosity for event confirmation screening. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). Default is 0.

cr_verbose:
Level of verbosity for conflict resolution analysis. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). A level of 2 is particularly useful when trying to understand why an event was removed due to conflict resolution. The highest level, 4, is only intended for debugging. Default is 0.

loc_verbose:

Level of verbosity for printed locator output. The level of detail increases as the value increases by integer steps from 0 (no printed output) to 4 (all output printed). One will usually not want to look at specific locator output unless the user has some particular interest in the behavior of a single event location. Default is 0.

A.3 Shared Modules Parameters

A.3.1 Network Probability Parameters

residual_over_sigma_max:
Maximum value of the ratio of the residual of the network probability value to the estimated standard deviation. This parameter is used in both probability of detection tests (before and after location). The default is 3.0.

max_obs_net_prob:
Maximum number of observations above which no probability of detection test is applied. This value and the value of the *residual_over_sigma_max* parameter are used by both the pre-location and post-location probability of detection tests. The default is 10.

A.3.2 Location and Confirmation Module Parameters.

loc_conf_level:
Locator confidence level. This is the confidence level at which the location error ellipse is computed. The default is 0.90.

loc_fix_depth:
If this string is specified, the locator keeps the depth fixed. The default is TRUE.

chi_outlier:
Chi-square threshold value used in the post-location outlier analysis within the locator module. This value is used to determine whether an arrival is an outlier for a particular event and to discard it from the preliminary event if it is the worst outlier. The default is 0.99.

max_smajax:
Maximum permissible semi-major axis of the location error ellipse. This is one of the confirmation criteria for a preliminary event. The default is 1000.0.

req_num_of_defining_detections:
Minimum number of defining detections for an event to be confirmed. The default is 3.

weight_threshold:
Minimum "weight" of an event for it to be confirmed. This is compared to the sum of all weights for the defining observations forming the event (see *primary_time_weight*, *secondary_time_weight*, *array_azimuth_weight*, *array_slow_weight*, *3comp_slow_weight*, *3comp_azimuth_weight*). This weighted-count confirmation test is described by *Bratt et al.* [1991, 1994]. The default is 3.9.

primary_time_weight:

Weight assigned to arrival times for primary phases for the weighted-count event confirmation test [*Bratt et al.*, 1991, 1994]. The default is 1.0.

secondary_time_weight:
Weight assigned to arrival times for secondary phases for the weighted-count event confirmation test [*Bratt et al.*, 1991, 1994]. The default is 0.7.

array_azimuth_weight:
Weight assigned to array azimuths for the weighted-count event confirmation test [*Bratt et al.*, 1991, 1994]. The default is 0.5.

array_slow_weight:
Weight assigned to array slowness for the weighted-count event confirmation test [*Bratt et al.*, 1991, 1994]. The default is 0.5.

3comp_slow_weight:
Weight assigned to slowness from 3-component data for the weighted-count event confirmation test [*Bratt et al.*, 1991, 1994]. The default is 0.25.

3comp_azimuth_weight:
Weight assigned to azimuth from 3-component data for the weighted-count event confirmation test [*Bratt et al.*, 1991, 1994]. The default is 0.25.

A.3.3 Cluster Analysis Parameters

cluster_min_ndef:
Minimum number of defining phases a preliminary event must possess to be clustered. The clustering algorithm is not applied to smaller events. The default is 6.

cluster_min_pct_overlap:
Minimum ratio of overlap that must exist for an association set to be included in a cluster. Valid values are 0.0 to 1.0. The default is 0.80.

A.3.4 Association-based Conflict Resolution Parameters

master_tradeoff_weight:
Weighting factor which controls the relative weighting between event quality and goodness-of-fit in the quality measure. The default is 0.2.

event_likelihood_weight:
Weighting factor for the event likelihood in the event quality factor. The default is 1.0.

dissolved_event_weight:
Weighting factor for the term in the event quality factor that is concerned with the likelihood that an event will be dissolved. The default is 0.1.

ndef_not_likely_to_dissolve_event:
Number of defining data above which it is unlikely that removing 1 arrival will dissolve (destroy)

an event. The default is 6.

ndef_which_will_dissolve_event:
Number of defining data below which it is likely that removing 1 arrival will dissolve (destroy) the event. The default is 3.

ndef_weight:
Weighting factor for the number of defining data (ndef) component of the event likelihood measure. The default is 1.0.

smajax_weight:
Normalized weight applied to semi-major axis of error ellipse (smajax) component of the event likelihood measure. The default is 0.8.

dnear_weight:
Normalized weight applied to distance to nearest station component of the event likelihood measure. The default is 0.5.

probdet_weight:
Normalized weight applied to event probability of detection component of the event likelihood measure. The default is 0.7.

ndef_no_confidence_bound:
Number of defining data below which we ascribe a very low confidence in the preliminary event. The default is 3.

ndef_high_confidence_bound:
Number of defining data above which we ascribe a very high degree of confidence in the preliminary event. The default is 10.

smajax_no_confidence_bound:
Size of semi-major axis of error ellipse (km) beyond which we ascribe a very low confidence in the preliminary event. The default is 500.0.

smajax_high_confidence_bound:
Size of semi-major axis of error ellipse (km) below which we ascribe a very high degree of confidence in the preliminary event. The default is 10.0.

dnear_no_confidence_bound:
Distance to nearest station (deg) beyond which we ascribe a very low confidence in the preliminary event. The default is 90.0.

dnear_high_confidence_bound:
Distance to nearest station (deg) below which we ascribe a very high degree of confidence in the preliminary event. The default is 10.0.

probdet_no_confidence_bound:
Network probability of detection, measured as residual/sigma, above which we ascribe a very low confidence in the preliminary event. The default is 3.0.

probdet_high_confidence_bound:
Network probability of detection, measured as residual/sigma, below which we ascribe a very high degree of confidence in the preliminary event. The default is 1.0.

A.4 Sample Parameter files

Sample GAassoc parameter file:

```
vendor="oracle"
database="t:machine:two_task"
account="account_name/password"
maxrecs=200000
in-arrival-table="arrival"
in-assoc-table="assoc"
in-origin-table="origin"
in-origerr-table="origerr"
out-origin-table="origin_ga"
out-assoc-table="assoc_ga"
out-origerr-table="origerr_ga"
minimum_ml_previously_determined=2.5
atten_file="slowamp.P"
table_path="/data/tab"
mb_dist_depth_suffix="pfact"
input_path="/home/ga/SDG"
input_file="GSETT3.spacing3.sector.-180deg.to.180deg"
num_first_sta=5
count_limit=10000
forward_transformation_list="(P PKPdf Pdiff Pn S ScP PKPab PKPbc PP ScS),(S Rg Sn
Lg),(Pn P Pg Pdiff S ScS),(Lg Sn Rg),(Sx Sn Lg Rg S ScS),(Tx PcP PKPbc PKPab),(Rg
Lg),(Sn Lg S)"
phases="Pg,Pn,P,Pdiff,PKPdf"
primary_phases="P,PKPdf,Pn,Pg,Pdiff"
freeze_arrivals_at_beam_points
primary_required_for_secondary
regional_S_phases="Sn,Lg,Rg,Sx"
sigma_time=3.
sigma_slowness=3.
start_time=789004800.
end_time=789033600.
chi_limit=.99
#
#optional processes parameters
#
#probdet_before_location
redundancy_required
probdet_after_location
do_clustering
```

```
do_association_based_conflict_resolution
#
#location parameters
#
loc_conf_level=0.90
loc_verbose=0
loc_fix_depth
chi_outlier=.99
#
# Event confirmation criteria
#
max_smajax=500.0
residual_over_sigma_max=3.
max_obs_net_prob=10
req_num_of_defining_detections=3
weight_threshold=3.9
primary_time_weight=1.0
secondary_time_weight=1.0
array_azimuth_weight=0.25
array_slow_weight=0.25
3comp_slow_weight=0.0
3comp_azimuth_weight=0.0
#
# Cluster analysis parameters
#
cluster_min_ndef=6
cluster_min_pct_overlap=0.8
#
# Association-based conflict resolution parameters
#
master_tradeoff_weight=0.2
event_likelihood_weight=1.0
dissolved_event_weight=0.1
ndef_not_likely_to_dissolve_event=6
ndef_which_will_dissolve_event=3
ndef_weight=1.0
smajax_weight=0.8
dnear_weight=0.5
probdet_weight=0.7
ndef_no_confidence_bound=3
ndef_high_confidence_bound=10
smajax_no_confidence_bound=500.0
smajax_high_confidence_bound=10.0
dnear_no_confidence_bound=90.0
dnear_high_confidence_bound=10.0
probdet_no_confidence_bound=3.0
probdet_high_confidence_bound=1.0
```

Sample GAconflict parameter file:

```
vendor="oracle"
database="t:machine:two_task"
account="account_name/password"
maxrecs=200000
in-arrival-table="arrival_ga"
in-assoc-table="assoc_ga"
in-origin-table="origin_ga"
in-origerr-table="origerr_ga"
out-origin-table="origin_final"
out-assoc-table="assoc_final"
out-origerr-table="origerr_final"
atten_file="slowamp.P"
table_path="/data/tab"
mb_dist_depth_suffix="pfact"
phases="Pg,Pn,P,Pdiff,PKPdf"
primary_phases="P,PKPdf,Pn,Pg,Pdiff"

start_time=789004800.
end_time=789033600.
#
#optional processes parameters
#
redundancy_required
do_clustering
do_association_based_conflict_resolution
#
#location parameters
#
loc_conf_level=0.90
loc_verbose=0
loc_fix_depth
chi_outlier=.99
#
# Event confirmation criteria
#
max_smajax=500.0
residual_over_sigma_max=3.
max_obs_net_prob=10
req_num_of_defining_detections=3
weight_threshold=3.9
primary_time_weight=1.0
secondary_time_weight=1.0
array_azimuth_weight=0.25
array_slow_weight=0.25
3comp_slow_weight=0.0
3comp_azimuth_weight=0.0
```

```
#
# Cluster analysis parameters
#
cluster_min_ndef=6
cluster_min_pct_overlap=0.8
#
# Association-based conflict resolution parameters
#
master_tradeoff_weight=0.2
event_likelihood_weight=1.0
dissolved_event_weight=0.1
ndef_not_likely_to_dissolve_event=6
ndef_which_will_dissolve_event=3
ndef_weight=1.0
smajax_weight=0.8
dnear_weight=0.5
probdet_weight=0.7
ndef_no_confidence_bound=3
ndef_high_confidence_bound=10
smajax_no_confidence_bound=500.0
smajax_high_confidence_bound=10.0
dnear_no_confidence_bound=90.0
dnear_high_confidence_bound=10.0
probdet_no_confidence_bound=3.0
probdet_high_confidence_bound=1.0
```

Distribution List

AFTAC/TTR	Technical Report (2 copies)
AFTAC/TTS	Technical Report (1 copy)
CA/STINFO	Technical Report (2 copies)